2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

A Topology-based Path Similarity Metric and Its Application to Sampling-based Motion Planning

Jory Denny, Kaiwen Chen, and Hanglin Zhou

***Abstract*— Many applications of robotic motion planning benefit from considering multiple homotopically distinct paths rather than a single path from start to goal. However, determining whether paths represent different homotopy classes can be difficult to compute. We propose metrics for efficiently approximating the homotopic similarity of two paths are, instead of verifying homotopy equivalence directly. We propose two metrics: (1) a naive application of local planning, a common subroutine of sampling-based motion planning, and (2) a novel approach that reasons about the topologically distinct portions of the workspace that a path visits. We present three applications of our metric to demonstrate its use and effectiveness: extracting topologically distinct paths from an existing roadmap, comparing paths for robot manipulators, and improving the computational efficiency of an existing sampling-based method, Path Deformation Roadmaps (PDRs), by over two orders of magnitude. We explore the trade-off between quality and computational efficiency in the proposed metrics.**

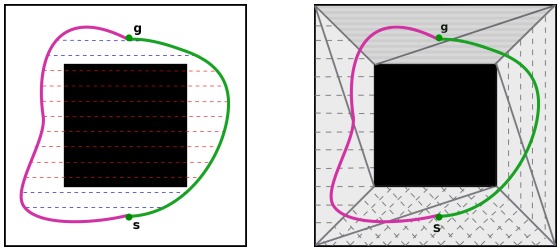
I. INTRODUCTION

Motion planning is an important problem in many domains such as robotics, bioinformatics, virtual reality, and virtual prototyping [16]. These applications often involve planning for multi-agent systems [3], dynamically changing environments [28], or planning under uncertainty [14], where having many candidate solutions to the problem is beneficial. Ideally, the candidate paths are different in nature and explore varying portions of the problem space. Differing paths are caused by obstacles in an environment. For example, a support pillar in the middle of a room creates two types of paths — those that go left and those that go right (Figure 1).

The set of paths that are equivalent in this sense belong to the same *homotopy class* [10]. Prior work has utilized homotopy classes for solving motion planning problems [4], [9], [11] or has focused on identifying homotopy classes [1], [2]. Determining if two solution paths are homotopy equivalent requires proving whether a continuous function that morphs one path into another does, or does not, exist — an operation that is potentially difficult to compute.

We are concerned neither with the process of finding the candidate paths nor with proving homotopy equivalence. Instead, we propose two approximation techniques (Figure 1) for determining whether two paths might be homotopy equivalent: (1) *Local Planner Success Similarity* (LPSS), which uses visibility [13], [27] to analyze a candidate deformation function that morphs one path into another, and (2) *Topology-Based Edit Distance Similarity* (TBEDS), which compares the regions of the workspace visited by each path

Jory Denny, Kaiwen Chen, and Hanglin Zhou are with the Spider Robotics Lab (SpiRoL), Department of Mathematics and Computer Science, University of Richmond, VA, USA, jdenny@richmond.edu.



(a) Local Planner Success Similarity (LPSS)

(b) Topology-Based Edit Distance Similarity (TBEDS)

Fig. 1. Example path comparison using the proposed similarity metrics.

to give a topologically informed likelihood for whether or not a deformation function exists. We demonstrate how these approximation techniques can be applied in sampling-based motion planning. Our specific contributions include:

- a definition and discussion of path similarity,
- the presentation of two novel similarity metrics: (1) local planner success similarity and (2) topology-based edit distance similarity, and
- demonstration of the technique to extracting distinct paths from an existing roadmap, comparing paths for robot manipulators, and improving the efficiency of a known sampling-based method, Path Deformation Roadmaps (PDRs), by over two orders of magnitude.

We consider our metrics complementary and explore the trade-off between the accuracy of the metric and its computational efficiency.

II. PRELIMINARIES

We define and review relevant terms for understanding our problem and metrics. Additionally, we highlight important related work.

A. Motion Planning

In this paper, we consider holonomic robots with d *degrees of freedom* (DOFs). DOFs parameterize a unique placement of the robot (center of mass position, orientation, joint angles, etc.) in its two- or three-dimensional world, or *workspace*. A configuration $q = \langle x_1, x_2, \ldots, x_d \rangle$ is a specification of the values for the DOFs, where x_i is the ith DOF. The set of all possible configurations is called the *configuration space* ($\mathcal{C}_{space}$) [18], and it can be partitioned into two main subsets: *free space* ($\mathcal{C}_{free}$) and *obstacle space* ($\mathcal{C}_{obst}$). These two subsets represent all possible valid, e.g., collision-free, and invalid configurations respectively. In general, it is infeasible to explicitly compute a representation of $\mathcal{C}_{obst}$ [24].

978-1-5386-8094-0/18/$31.00 ©2018 IEEE

Although it is not the focus here, we review an example motion planning approach, Probabilistic RoadMaps (PRMs) [15], in order to give an understanding of how solution paths can be found and given as input into our metrics. PRMs construct a map of $\mathcal{C}_{free}$ by first randomly sampling valid configurations. Nearby samples are then connected by validating simple paths between them, which form the edges of the map. A *local planner*, e.g., straight-line interpolation, is an algorithm used to verify these transitions. Finally, start and goal configuration pairs are connected to the roadmap and a graph search, e.g., A*, is used to extract a solution.

We define a *path*, Definition 1, as a sequence of adjacent configurations. When comparing two paths of differing lengths, there could be a large difference in numbers of configurations in each sequence. To facilitate this representational issue, we use a secondary definition of paths based on a parametric representation of the trajectories. t-resolution paths, Definition 2, are a sequence of configurations along a path discretized at resolution steps in the parametric form. Due to this, two t-resolution paths will have the same number of configurations in each sequence, and we can carefully choose the resolution to be appropriate for use in our metrics. These definitions are similar to those found in path smoothing and modification [8].

Definition 1: A **path** $\pi = \{q_1, q_2, \ldots, q_m\}$ is a contiguous sequence of configurations. Two configurations q_i and q_{i+1} are adjacent if $\delta(q_i, q_{i+1}) \leq r$, where δ is a distance function and r is a problem specific resolution.

Definition 2: Let $\pi = \{q_1, q_2, \ldots, q_m\}$ be a path. The **t-resolution path** τ of π is a contiguous sequence of configurations discretized at t_{res} steps, where t_{res} is a resolution, from a parametric representation $t \in [0, 1]$ of π, where $t = 0$ is q_1, $t = 1$ is q_m. Let τ^k denote the kth configuration of τ, where $k \in [0, 1]$.

B. Topology

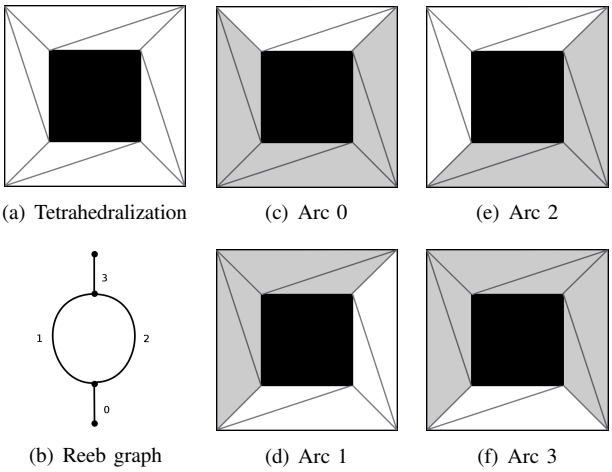
In this section, we review important aspects of topology related to our study: homotopy classes and Reeb Graphs.

1) Homotopy: Definition 3 presents the concept of homotopy equivalence [10] — loosely, it is the concept that two paths visit the same parts of $\mathcal{C}_{space}$. Further, Definition 4 shows that homotopy classes [10] are a set of pairwise homotopy equivalent paths. Homotopy classes group portions of the space together in a semantic sense. As an example, consider a fork in a road. There is in general an infinite set of paths and combination of footstep placements along the paths, but they can be grouped into two classes: those that take the left path and those that take the right path.

Definition 3: Two paths $\pi_1, \pi_2 \in \mathcal{C}_{free}$ (or t-resolution paths τ_1 and τ_2) are **homotopy equivalent** if and only if a function exists that continuously deforms π_1 to π_2 in $\mathcal{C}_{free}$ (τ_1 to τ_2), i.e., without transitioning through $\mathcal{C}_{obst}$.

Definition 4: A **homotopy class** is a set of paths Π (or t-resolution paths) such that all pairs $\pi_i, \pi_j \in \Pi$ are homotopy equivalent.

These concepts are fundamental to motion planning and define the basis of many methods. As an example, geometric methods [4], [9], [11] use mathematical reasoning to represent homotopy classes and find optimal planning solutions but are inefficient for many planning problems.



(a) Tetrahedralization (c) Arc 0 (e) Arc 2

(b) Reeb graph (d) Arc 1 (f) Arc 3

Fig. 2. Example computation of a Reeb graph from a tetrahedralization. (a) Tetrahedralization of an environment. (b) Reeb graph formed from the tetrahedralization. (c-f) The tetrahedrons that are mapped to each Reeb Arc.

Certain sampling-based methods are topologically inspired. Visibility-based PRMs [27] represent the space using a few "well placed" configurations. In this way, this method creates roadmaps that guarantee coverage and connectivity in at least one homotopy class. This method was extended to preserve homotopy classes [25] by checking not only visibility constraints but also merging portions of a roadmap that map to the same homotopy class. This class of methods attempts to introduce and define *useful cycles* into PRMs. Path Deformation Roadmaps (PDRs) further reduces the size of roadmaps based on this concept [13]. This methodology uses similar, but distinct, techniques to the first metric we propose. Other approaches have discussed and incorporated useful cycles [6], [8], [20], [28].

Algorithms have been proposed that explicitly identify and represent homotopy classes using sophisticated mathematical reasoning [1], [2]. The aim of these techniques is usually to find optimal solutions or paths constrained within a specific homotopy class. These methods could themselves inform a more expensive metric of comparing homotopy equivalence in the future. As such, they are considered complementary to our proposed work.

2) Reeb Graphs: Concise representations of all homotopy classes of a topological space exist. One such representation, a Reeb Graph [23] represents transitions in level sets of a real-valued function on a manifold. The nodes of a Reeb Graph encode critical values of the function, referred to as a Morse function, and the edges, or Reeb Arcs, are the topological transitions between them. For example, Figure 2(b) is a Reeb Graph for the space shown in Figure 2(a), where the y-value of any point is the value of the Morse function. Reeb Graphs have been applied to numerous domains and problems in computational geometry and computer graphics such as shape matching [12], iso-surface remeshing [30],

simplification [29], and sampling-based planning [5].

To compute a Reeb Graph from a workspace, we use the algorithm found in [21]. Other methods have since improved on the complexity and could be alternatively used [7]. This algorithm begins by computing a Delaunay tetrahedralization of the free workspace [26] (gray in Figure 2(a)). A Reeb Graph is initialized from the 1-skeleton of the tetrahedralization (vertices and edges of the tetrahedrons form nodes and Reeb Arcs respectively). Then, the 2-skeleton of the tetrahedralization (facets of the tetrahedrons) is used to reduce the initial graph to be a true Reeb Graph that encodes topological information of the space. For all triangles of the 2-skeleton, the algorithm merges the three associated Reeb Arcs into two. A triangle is a safe criterion for merging as it represents a sector of topological equivalence. During post-processing, 2-nodes of the Reeb Graph are removed by merging the two associated Reeb Arcs into one. Figure 2(b) shows the resulting Reeb Graph from the tetrahedralization (Figure 2(a)). From our experience, a Reeb Graph from environments composed of over $2,000$ facets can be computed on the order of a few seconds.

It is important to note that 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 — a critical point represents a divide in the free workspace volume. Thus, each Reeb Arc captures a distinct workspace homotopy class between two critical points.

During construction, related tetrahedra to Reeb Arcs are tracked. The related tetrahedra for each Reeb Arc in our example are shown in Figures 2(c)–2(f). It is important to observe that tetrahedrons are not unique to each Reeb Arc; they may correspond to more than one. A good example is that a tetrahedron related to a critical point is shared among all Reeb Arcs connected to the critical point.

III. ALGORITHMS FOR COMPUTING PATH SIMILARITY

We present the problem of approximating path similarity and explore the requirements for a high quality metric. After, we describe and discuss two similarity metrics that we consider to be complementary. The first is based on a naive application of local planning, a common subroutine of sampling-based motion planners, and the second is rooted in a topological analysis of the paths in workspace.

A. Overview of Path Similarity

As discussed in Section II, all paths between a set start configuration and goal configuration pair can be classified into equivalence classes (i.e., homotopy classes). Determining if two paths are in the same homotopy class can be difficult for the general case. Instead, we are interested in defining a metric that approximates the likelihood that two paths are homotopy equivalent.

We define a *path similarity metric*, μ, as a function that takes as input two t-resolution paths and returns a real number in the range $[0, 1]$. For a high-quality metric, a zero value implies the paths are in distinct homotopy classes, and a value of one implies that the paths are homotopy equivalent. All values in between define the likelihood that the paths are in (or not in) the same homotopy class.

B. Local Planner Success Similarity

We introduce a naive similarity metric based upon application of local planning, a common subroutine of sampling-based motion planners [15]. The metric, called Local Planner Success Similarity (LPSS), is shown in Algorithm 1. The metric essentially computes a ratio of successful local planning attempts to total attempts between configurations of the two paths with the same t value. In Figure 1(a), we show two example paths (green and magenta) and the local planning attempts between pairs of configurations along the paths (blue dashed lines are successful attempts, and red dashed lines are unsuccessful attempts).

Algorithm 1 Local Planner Success Similarity

Input: t-resolution Paths τ_1, τ_2, Local planner Δ
Output: Similarity $s \in [0, 1]$

```
s ← 0, n ← 0
for t ∈ [0, 1] at t-resolution increments do
    if Δ(τ1^t, τ2^t) then
        s ← s + 1
    n ← n + 1
return s/n
```

In LPSS, it is easy to verify that a metric value equal to one confirms a linear transformation of one path into another up to a t-resolution, i.e., they are in the same homotopy class. A value close to one implies that a transformation is likely to exist, but the linear transformation is not it. A value close to zero implies that a transformation is unlikely to exist that morphs one path into another, i.e., the paths are in distinct homotopy classes. We note that this algorithm will not give a value of zero because the start and goal configurations of the paths are identical, i.e., a local plan between them is trivially successful.

By only verifying one possible transformation, LPSS does not specifically use any topological information in determining similarity. However, a strength of the metric is that when two paths are determined to be similar, we not only know the equivalence class but also a function that deforms one path into the other.

This method is similar to the deformation technique in [13]. However, that work specifically uses an A* search through the pair-wise visibility of configurations between two t-resolution paths. Our method is a more efficient algorithm and returns a numeric value.

C. Topology-based Edit Distance Similarity

We present a novel similarity metric rooted in topological reasoning. Essentially, the metric compares the portions of the workspace each path visits. Our metric, called Topology-Based Edit Distance Similarity (TBEDS), shown in Algorithm 2, translates each path into a string and uses the notion

Algorithm 2 Topology-based Edit Distance Similarity

Input: t-resolution Paths τ_1, τ_2, Local planner Δ
Output: Similarity $s \in [0, 1]$

$s_1 \leftarrow$ TOPOLOGYBASEDSTRING(τ_1)
$s_2 \leftarrow$ TOPOLOGYBASEDSTRING(τ_2)
$d \leftarrow$ LEVENSHTEINDISTANCE(s_1, s_2)
return $1 - \frac{d}{\max(|s_1|,|s_2|)}$

Algorithm 3 Path to Topology-based String

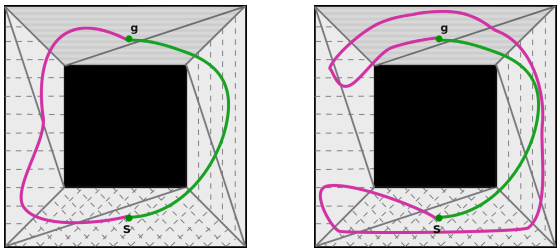
Input: t-resolution Path τ
Output: String s

{Find tetrahedron sequence that τ passes through}
Sequence $tetras \leftarrow \emptyset$
for $t \in [0, 1]$ at t-resolution increments **do**
  $tetras \leftarrow tetras \cup$ FINDTETRAHEDRON(τ^t)
UNIQUE$(tetras)$
{Find sequence of Reeb arcs that τ passes along}
Sequence $reebArcs \leftarrow \emptyset$
for all $tetra \in tetras$ **do**
  $reebArcs \leftarrow reebArcs \cup$ FINDREEBARCS$(tetra)$
UNIQUE$(reebArcs)$
return REEBARCSEQUENCETOSTRING$(reebArcs)$

of edit distance as the metric [19]. Specifically, we use Levenshtein Distance [17] in our implementation. Levenshtein Distance counts the minimum number of additions, deletions, and substitutions required to convert one string into another. The string used for each path is the sequence of workspace regions that the path traverses. Edit distance has been used in motion planning in the context of path merging [22] — a tangential problem to path similarity.

To determine the sequence of workspace regions, we use the methodology shown in Algorithm 3. The method assumes that the workspace of the robot has been tetrahedralized and converted into a Reeb Graph, as discussed in Section II. The first phase of the algorithm determines the sequence of tetrahedrons that a point of interest of the robot passes through along the path. The point of interest could be something as simple as the center of mass of the robot or more complex like the end-effector location of a manipulator robot. After the initial sequence is determined, we run an algorithm called UNIQUE on the sequence. This eliminates adjacent duplicates from the sequence. In the second phase of the algorithm, we map the tetrahedron sequence to a sequence of sets of Reeb Arcs. For each tetrahedron, we determine all of the associated Reeb Arcs to it and use this as the set — recall from Section II that based on the reduction algorithm tetrahedrons may belong to more than one Reeb Arc. Again, we use UNIQUE to eliminate adjacent duplicates from the sequence of Reeb Arc sets. Finally, we determine a string by mapping each unique set of Reeb Arcs to a unique "character." We use the term "character" loosely, and more specifically map each set to a non-negative integer.

As an example of converting a path to a string using Algorithm 3, we refer to the right path of Figure 3(a)



(a) Typical case (b) Degenerate case

Fig. 3. Example computations of converting paths to topology-based strings. Unique Reeb Arc sets are shown as distinct shaded regions. (a) Typical case of representing a simple path that visits a linear sequence of Reeb Arc sets. (b) Atypical (degenerate) case of representing a complex path that visits a backtracking sequence of Reeb Arc sets.

(green). Looking at this path, we first observe that a point moving along this path at a t-resolution visits a sequence of tetrahedrons. Second, we can determine the unique sequence of Reeb Arc sets traversed along the path. The gray patterns show the unique Reeb Arc sets in this environment — visually, the reader can refer to Figure 2 to view the tetrahedrons for each Reeb Arc and verify the unique Reeb Arc set from their overlaps. So, this path visits the bottom region, then the right region, and then the top region. We translate this to a unique string "BRT." Following the same observations, we can see the left path (magenta) translates to the string "BLT." After converting the paths to strings, we can apply Algorithm 2 to see that the edit distance is one and the TBEDS metric is $\frac{2}{3}$ — we can substitute 'R' for 'L' to get the second string and a Levenshtein Distance of one.

In TBEDS, if a value of one is returned, then the paths taken by the point of interest of the robot are in the same homotopy classes of the workspace. This may or may not imply the same for the configuration space paths; it depends on the specific problem and choice of the point of interest. When the metric returns a value close to zero, the paths taken by the point of interest are in disjoint homotopy classes of the workspace and therefore are in disjoint homotopy classes of the free configuration space. The metric cannot return zero because the start and goal for the paths are identical.

The TBEDS metric provides a few interesting discussion points from a theoretical view. First, the metric is rooted in topological reasoning, an important distinction from the nature of LPSS. We established a metric that analyzes the regions of the workspace (preprocessed and grouped into topological equivalences) that a path visits. TBEDS is limited to reasoning about workspace regions and cannot reason about regions of $\mathcal{C}_{free}$.

Second, there are degenerate cases for TBEDS. Specifically, we refer to the example shown in Figure 3(b). The simple path (green) is identical to the right path in the previous example and translates to "BRT." Analyzing the second complex path (magenta), we note it translates to "BLBRTLT." The Levenshtein edit distance is now 5 (one substitution and four deletions) and our TBEDS metric is $\frac{2}{7}$. This is problematic given that the two paths are homotopy equivalent. One method to overcome this is to apply path

smoothing to improve the paths before comparing them; however, some path smoothing methods [8], like shortcutting, can actually change the homotopy class of the path. Because of this, we did not apply smoothing in this study. In our applications, the degenerate case did not arise often, and we leave further exploration of them to future work.

IV. APPLICATION: PATH EXTRACTION

We apply our metrics to the application of extracting homotopically distinct paths from a roadmap. Further, we explore the trade-offs of each metric in more topologically complex environments, i.e., they contain more homotopy classes.

A. Algorithm Overview

For our application, we explore the problem of extracting topologically distinct paths from a roadmap. Algorithm 4 is used for this purpose. It is a modification of depth-first search that explores all simple paths between a start and goal node of a graph. For each path that is found, the algorithm uses a path similarity metric to compare it to all previously found paths. If the new path is significantly different (compared to a threshold), then the new path is added to the output set.

This algorithm has exponential complexity, and we use it on small roadmaps only. Note, this algorithm is not the focus of our paper, and we chose this approach to extensively test and compare our metrics.

B. Analysis

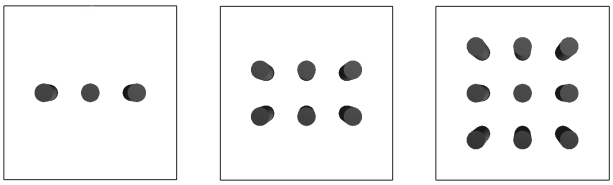
We implemented our metrics in C++ using the GNU gcc compiler version 5.4.0, and we ran all experiments on Ubuntu 16.04 with an Intel® Core™ i7-6700U CPU at 3.4 GHz and 16 GB of RAM.

We used a t-resolution of 0.01 and the Euclidean distance as the distance function where needed. LPSS used a straight-line local planner using a bisection evaluation order over the interpolation.

Algorithm 4 Generate Paths

Input: Roadmap R, t-resolution Paths Π, Threshold $threshold$,
Vertex u, Vertex $goal$, Vertex sequence p

```
1: u.VISITED(true)
2: p ← p ∪ {u}
3: if u = goal then
4:    t-resolution path τ ← GETPATH(R, p)
5:    for all τ_i ∈ Π do
6:       if PATHSIMILARITY(τ, τ_i) > threshold then
7:          return
8:    Π ← Π ∪ {τ}
9: else
10:   for all v ∈ u.ADJACENTVERTICES() do
11:      if ¬v.ISVISITED() then
12:         GENERATEPATHS(R, Π, v, g, p, threshold)
13: u.VISITED(false)
```



(a) `Cyl3` (b) `Cyl6` (c) `Cyl9`

Fig. 4. Environments for the path extraction application. The environments are composed of increasing numbers of columns that create topologically more complex scenes in the workspace, i.e., `Cyl9` (c) has more homotopy classes than `Cyl3` (a). The query is from the bottom to the top in each environment.

TABLE I

REEB GRAPH SIZES, AVERAGE ROADMAP SIZES, THRESHOLDS, AND AVERAGE NUMBER OF PATHS EXTRACTED FOR THE EXPERIMENTAL ANALYSIS.

	`Cyl3`	`Cyl6`	`Cyl9`
Reeb Graph Sz. (nodes, edges)	8, 10	14, 19	20, 28
Avg. Roadmap Sz. (nodes, edges)	14, 53	17, 57	20, 56
Threshold – TBEDS	0.50	0.50	0.50
Threshold – LPSS	0.85	0.75	0.70
Avg. Num. Unique Paths – TBEDS	6.6	6.8	6.7
Avg. Num. Total Paths – TBEDS	1400	1270	577
Avg. Num. Unique Paths – LPSS	6.4	6.0	7.6
Avg. Num. Total Paths – LPSS	1060	1150	617

We analyze the algorithm using the environments shown in Figure 4. The environments contain approximated cylinders of radius 2m and height 6m in an environment that is $20 \times 20 \times 6$m. The environments increase in topological complexity (contain more homotopy classes) as the number of cylinders increase from three to nine. They are arranged as uniform rows, so the first environment has one row of three cylinders and the last environment has three rows of three cylinders. We used a query from the bottom to the top in each environment.

We first create roadmap using the PRM approach [15] that captures some or all of the homotopy classes. The roadmap nodes were generated by performing uniform random sampling, where samples are generated a predefined distance apart, and the edges were formed by a radius-based neighborhood connection scheme using a straight-line local planner. The size of the roadmaps were limited to accommodate the application. We created 30 different roadmaps for the experiments. Table I shows the average roadmap and Reeb Graph size for each environment.

For the experiment, we selected thresholds so that we yield about the same total number of paths compared and about the same number of unique paths extracted. To verify, we present the thresholds and the average number of paths extracted using each method in Table I. All averages are very similar and within one standard deviation from each other. The threshold for LPSS was reduced with increasing environment complexity as the metric becomes more accurate with increased $\mathcal{C}_{obst}$ volume.

Figure 5 presents the average time taken to analyze the roadmaps for path extraction over the 30 trials. As seen, the TBEDS metric is about ten times faster across the three

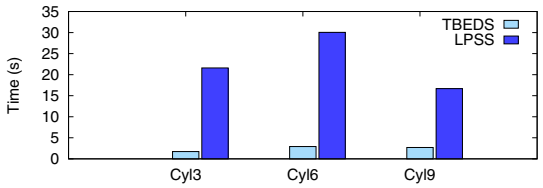
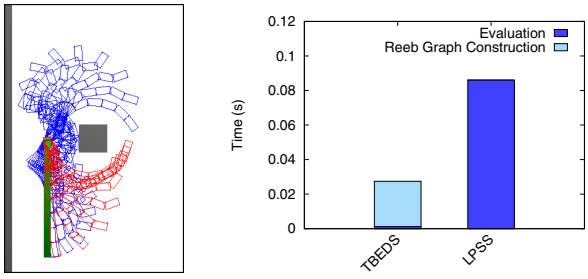


Fig. 5. Average time shown for path extraction over 30 trials using both the LPSS and TBEDS metrics in all three cylinder environments.



(a) `Manipulator` (b) Time Comparison

Fig. 6. Experimental comparison of TBEDS and LPSS metrics on a robot manipulator. (a) Environment with fixed-base manipulator arm. The two paths used in the comparison are shown in red and blue. (b) Timing results for the comparison.

environments. Using a statistical t-test TBEDS is faster than LPSS with a p-value < 0.0001 across all three environments. The largest computation cost of TBEDS is the Reeb Graph construction, but its cost is amortized over the execution of the algorithm.

TBEDS and LPSS compute metrics of different topological meaning, but from this experiment we also see a significant trade-off in computational efficiency.

V. APPLICATION: ROBOT MANIPULATORS

We apply our metrics to comparing paths of a robotic arm. Our system setup is identical to Section IV-B. The key difference is that we use the position of the end effector at the point-of-interest of the robot for TBEDS.

We use the scenario shown in Figure 6(a). In this problem, an eight DOF manipulator arm must reach around a cube, while avoiding collision with a wall located to the left of the robot. The robot's base is fixed to the origin of the environment. We computed two different paths to the same end effector position using a sampling-based planner, one that goes below the obstacle (red) and the other that reaches above the obstacle (blue). We analyze each metric on the pair of homotopically distinct paths of the end effector position and compare both the similarity computed and the computation time taken. The analysis was repeated 30 times and the average computation time is shown in Figure 6(b).

Both TBEDS and LPSS present similarity values noting large differences between the two paths, 0.38 and 0.40 respectively. Timing is shown in Figure 6(b). TBEDS is about four times faster than LPSS. Again the Reeb Graph construction time is dominant. Considering evaluation time only, TBEDS is approximately seven times faster.

Algorithm 5 Path Deformation Roadmap

Input: Maximum failure iterations n_{max}
Output: Path Deformation Roadmap R

$n_{fail} \leftarrow 0$
Roadmap $R \leftarrow \emptyset$
while $n_{fail} < n_{max}$ **do**
  $q \leftarrow$ GETRANDOMFREECFG()
  $R_{vis} \leftarrow$ VISIBLESUBROADMAP(q)
  if $R_{vis} = \emptyset$ **then**
    ADDNEWGUARD(R, q)
    $n_{fail} \leftarrow 0$
  else if R_{vis}.NUMCCS() > 1 **then**
    $\{v_1, v_2\} \leftarrow$ TWONEAREST(R_{vis}, q)
    $\tau \leftarrow$ PATHINDUCED(v_1, q, v_2)
    if ISUSEFULCYCLE(R, v_1, v_2, τ) **then**
      ADDCYCLICPATH(R, τ)
      $n_{fail} \leftarrow 0$
    else
      $n_{fail} \leftarrow n_{fail} + 1$
  else $\{R_{vis}$.NUMCCS() $= 1\}$
    $n_{fail} \leftarrow n_{fail} + 1$
return R

VI. APPLICATION: PATH DEFORMATION ROADMAPS

We use our metrics to improve on an existing technique in sampling-based motion planning. The methodology, called Path Deformation Roadmaps (PDRs) [13], extends Visibility-based PRM [27] to contain useful cycles in the roadmap that represent multiple homotopy classes in an environment. We review the core algorithm, explain how our metrics can be used in the approach, and experimentally analyze them.

A. Algorithm Overview

We present an iterative formulation of PDR construction in Algorithm 5. In each iteration, a random configuration $q \in \mathcal{C}_{free}$ is generated and the visible portion of the roadmap R_{vis} to q is determined — this is computed incrementally by performing either rigorous visibility checks, i.e., local planner attempts, or by reasoning about the clearance (minimum visibility) of configurations, which we used here (see [13] for further implementation details). If R_{vis} is empty, q is classified as a new guard [27], a configuration covering a new part of $\mathcal{C}_{free}$, and added to the roadmap. If R_{vis} contains multiple connected components, q becomes a candidate for creating a useful cycle in the roadmap. The algorithm finds the two nearest configurations in R_{vis} to q, induces a cycle, and checks the cycle for redundancy (Algorithm 6). If R_{vis} is one connected component, or q does not create a useful cycle, the number of failed iterations is incremented. The algorithm continues until a maximum number of failed iterations is reached.

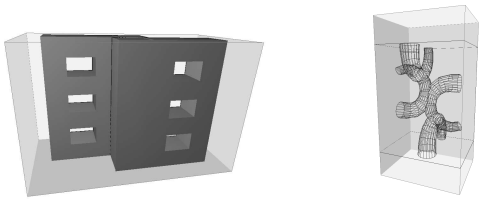
The original PDR algorithm uses an A* search through pair-wise visibility checks of two t-resolution candidate paths. A successful search yields a path deformation, and thus a homotopically redundant cycle is determined.

Algorithm 6 IsUsefulCycle

Input: Roadmap R, configurations v_1, v_2, and candidate path τ. Pre-configured input for number of comparison paths k, path similarity metric μ, and threshold ϵ.

Output: **true** if τ augments R with a useful cycle between v_1 and v_2, **false** otherwise

```
T ← KSHORTESTPATHS(R, k)
for all τ' ∈ T do
    if μ(τ, τ') >= ε then
        return false
return true
```



(a) `Building` (b) `Tunnel`

Fig. 7. Additional environments for application of the metrics to creating PDRs. (a) `Building` and (b) `Tunnel` both contain inherently three-dimensional homotopy classes providing increased difficult of roadmap creation.

We modify the PDR algorithm's verification for useful cycles — its most expensive step. Our updated approach is shown in Algorithm 6 which abstracts the path redundancy check with a path similarity metric, μ. We compare the metric value to a threshold for redundancy determination. In our experiments, TBEDS, LPSS, and A* refer to constructing a PDR using the TBEDS metric, LPSS metric, or original A* search, respectively, for checking useful cycles.

B. Analysis

In this experiment, we compare the cost of generating a roadmap using the PDR algorithm with TBEDS, LPSS, and A* as metrics for determining whether two paths are deformable. We use the same experimental setup as that used in Section IV-B.

For this application, we use environments similar to those shown in Figure 4 except the radii of the cylinders were increased to 4m and spaced evenly in the environment. Additionally, we use the environments in Figure 7, both of which have a three-dimensional workspace, a six-dimensional $\mathcal{C}_{space}$ (translation and rotation of a rigid body robot), and contain multiple homotopy classes. `Building`, Figure 7(a), represents a UAV robot (no kinematics are considered) inspecting an urban apartment complex that has a solid center and two wings with four stories each. `Tunnel`, Figure 7(b), is a maze of tunnels through complex geometry that exercises the cost of collision detection and Reeb Graph construction more fully.

Table II displays the variables used for the similarity metric thresholds and maximum failures allowed for for PDR construction. We used $k = 5$ shortest paths in the algorithm across all methods. We determined the values to create roadmaps of comparable sizes and qualities (bold) to facilitate a fair comparison between the various approaches. The average time to construct each PDR in 30 trials (10 in `Tunnel`) with outliers removed and normalized to the time TBEDS required is shown in Figure 8. In darker color, the time required specifically for the useful cycle check (Algorithm 6) is shown. A* did not complete within 10,000 seconds on any trial in the `Building` or `Tunnel` environments. The time reported for TBEDS includes the time required to construct the Reeb Graph representation of the environment.

Our results show that both TBEDS and LPSS allow PDRs to be more efficiently constructed by more than two orders of magnitude in certain environments compared with A*. Further, TBEDS is more efficient in time than LPSS by a factor of two to twelve depending on the environment.

The core reason this occurs is a reduction in the time taken for useful cycle validation (shown in dark color in the

TABLE II

SIMILARITY METRIC THRESHOLD AND MAXIMUM FAILED ITERATIONS USED FOR PDR CONSTRUCTION, AND THE RESULTING AVERAGE ROADMAP SIZES (NODES AND EDGES) OVER ALL TRIALS (OUTLIERS REMOVED) FOR TBEDS, LPSS, AND A*. NOTE, A* WAS UNABLE TO COMPLETE COMPUTATION IN 10,000 SECONDS ON ALL TRIALS IN BUILDING AND TUNNEL.

		TBEDS	LPSS	A*
`Cyl3`	Threshold	0.45	1.0	1.0
	Max Failure	40	200	200
	Roadmap Size	**12, 38**	**12, 33**	**9.1, 22**
`Cyl6`	Threshold	0.35	1.0	1.0
	Max Failure	40	45	45
	Roadmap Size	**24, 74**	**19, 48**	**17, 40**
`Cyl9`	Threshold	0.45	1.0	1.0
	Max Failure	40	30	30
	Roadmap Size	**39, 121**	**27, 70**	**24, 59**
`Building`	Threshold	0.30	1.0	1.0
	Max Failure	30	20	20
	Roadmap Size	**15, 31**	**19, 42**	–
`Tunnel`	Threshold	0.35	1.0	1.0
	Max Failure	600	500	500
	Roadmap Size	**31, 76**	**25, 58**	–

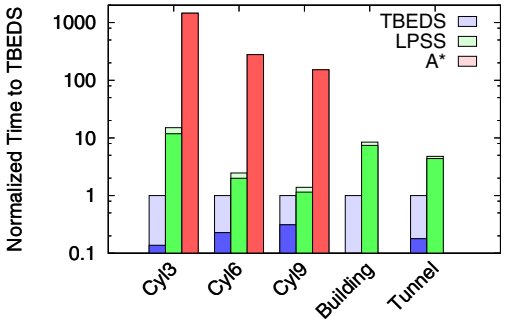


Fig. 8. Time for PDR creation with TBEDS, LPSS, and A* normalized to TBEDS in the five environments. In darker color is the time required for path comparison phase of the algorithm. Results averaged over 30 trials (10 in `Tunnel`, outliers removed). A* did not complete within 10,000 seconds on any trial in the `Building` or `Tunnel` environments.

histogram). Note that in both LPSS and A* the dominant computation is the useful cycle check, often taking more than 90% of the computation time. While LPSS reduces the number of collision checks using an approximated validation of deformation compared with A*, TBEDS entirely eliminates collision checking during this phase. Ultimately, the A* approach is not scalable to complicated environments. Using path similarity allows PDRs to be constructed in a reasonable time frame.

This is not without a trade-off. As noted before, LPSS performs a deformation validation check, whereas TBEDS simply approximates a likelihood of this existing. However, with a cheaper useful cycle check more effort could be spent checking more samples to create a higher quality roadmap. We would like to explore this effect and apply concepts from TBEDS to further reduce the computation time required for PDR construction.

VII. CONCLUSION

We presented the problem of approximating path similarity and proposed two similarity metrics. Further, we demonstrated the usefulness and efficiency of our metrics in various ways. We consider our metrics complementary in how they approximate path similarity.

In the future, we would like to explore how the notion of path similarity can be used in developing and analyzing novel motion planning algorithms and applications for multi-agent robot systems. Additionally, we would like to continue to improve upon the topology-based edit distance similarity to overcome degenerate cases.

REFERENCES

[1] S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based path planning with homotopy class constraints. In *Proc. of the Twenty-fourth AAAI Conference on Artificial Intelligence*, July 2010.

[2] S. Bhattacharya, M. Likhachev, and V. Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[3] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, volume 1, pages 540–545 vol.1, 2002.

[4] D. Demyen and M. Buro. Efficient triangulation-based pathfinding. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 942–947. AAAI Press, 2006.

[5] J. Denny, R. Sandström, A. Bregger, and N. M. Amato. Dynamic region-biased rapidly-exploring random trees. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, December 2016.

[6] A. Dobson and K. E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *Int. J. of Robotics Research*, 33:18–47, 2014.

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

[8] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *Int. J. of Robotics Research*, 26(8):845–863, 2007.

[9] D. Grigoriev and A. Slissenko. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, ISSAC '98, pages 17–24, New York, NY, USA, 1998. ACM.

[10] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[11] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. In *Proc. 2nd Workshop Algorithms Data Struct.*, volume 519 of *Lecture Notes Comput. Sci.*, pages 331–342. Springer-Verlag, 1991.

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

[13] L. Jaillet and T. Siméon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *Int. J. of Robotics Research*, 27(11-12):1175–1188, 2008.

[14] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 2013.

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

[16] J. C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *Int. J. of Robotics Research*, 18(11):1119–1128, 1999.

[17] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, Feb. 1966.

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

[19] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, Mar. 2001.

[20] D. Nieuwenhuisen and M. H. Overmars. Useful cycles in probabilistic roadmap graphs. pages 446–452, 2004.

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

[22] B. Raveh, A. Enosh, and D. Halperin. A little more, a lot better: Improving path quality by a path-merging algorithm. *IEEE Transactions on Robotics*, 27(2):365–371, April 2011.

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

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

[25] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of homotopy classes with probabilistic road map. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, volume 3, pages 2317–2322 vol.3, 2002.

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

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

[28] J. P. van den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars. Creating robust roadmaps for motion planning in changing environments. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 1053–1059, Aug 2005.

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

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