Topo-Geometrically Distinct Path Computation using Neighborhood-augmented Graph, and its Application to Path Planning for a Tethered Robot in 3D

Alp Sahin and Subhrajit Bhattacharya

***Abstract*—Many robotics applications benefit from being able to compute multiple geodesic paths in a given configuration space. Existing paradigm is to use topological path planning, which can compute optimal paths in distinct topological classes. However, these methods usually require non-trivial geometric constructions which are prohibitively expensive in 3D, and are unable to distinguish between distinct topologically equivalent geodesics that are created due to high-cost/curvature regions or prismatic obstacles in 3D. In this paper, we propose an approach to compute k geodesic paths using the concept of a novel neighborhood-augmented graph, on which graph search algorithms can compute multiple optimal paths that are topo-geometrically distinct. Our approach does not require complex geometric constructions, and the resulting paths are not restricted to distinct topological classes, making the algorithm suitable for problems where finding and distinguishing between geodesic paths are of interest. We demonstrate the application of our algorithm to planning shortest traversible paths for a tethered robot in 3D with cable-length constraint.**

***Index Terms*—Motion and Path Planning, Multi Path Planning, Graph Search-based Path Planning, Tethered Robot**

I. Introduction

A. Background and Motivation

The optimal path planning problem for robots requires a sequence of transformations to be found from an initial to a goal configuration that avoids obstacles and globally minimizes a cost functional. This classical instance of the problem is solved in lower dimensions by discrete graph-search algorithms [1, 2, 3, 4, 5, 6] in a systematic and complete manner up to the resolution of the discretization, whereas sampling-based path planning algorithms [7, 8, 9] provide a more effective solution for higher-dimensional problems, however, only with probabilistic completeness.

In this paper, we are interested in the k geodesic path planning problem, where the goal is to find k non-identical paths in the free configuration space that locally (i.e. under small perturbations of the path) minimize a cost functional. This instance of the problem has many practical applications in robotics including motion planning for multi-robot teams and tethered robots. Once a set of distinct geodesics are found, they can be used by a higher-level motion planning algorithm as reference paths to distribute a swarm of mobile robots along different routes in order to avoid congestion in the environment while keeping the overall travel times at a minimum [10]. In a similar manner, geodesic paths can serve as references for multiple robots [11] or a heterogeneous team of robots and humans to explore unknown or partially-known environments in search-and-rescue type operations [12]. Multiple geodesic paths can be provided to a user interface for interactive and transparent operation [13, 14] or input to an external algorithm for further assessment and decision-making regarding robot's motion. Geodesic paths can be used to represent taut tether configurations and enumerating them enables planning for tethered robot motion [15] or for robots carrying cables [16].

Topological path planning (TPP) methods can be used for computing optimal paths in distinct homotopy classes [17, 18, 19, 20]. However, these methods (a) require a priori geometric constructions, (b) require complex representations of the homotopy groups of the configuration space along with complicated equivalence check algorithms, (c) cannot identify multiple geodesic paths within the same homotopy class, and, (d) are not easily extended to 3 and higher dimensional configuration spaces. The limitations described in (c) is frequently encountered when a robot navigates on a surface with non-zero curvature or around prismatic obstacles without holes in spatial domains.

Other methods that compute multiple non-identical paths require global information regarding the configuration space, such as the visibility/line-of-sight between configuration points and the global topology of the configuration space [21, 22, 13]. However, such information can be challenging to define and compute in an automated fashion when the configuration spaces become more complex.

The topo-geometric path planning approach proposed in this paper provides a solution to the k geodesic path planning problem, while remaining applicable to higher dimensional and geometrically complex configuration spaces. This is accomplished by augmenting any wavefront propagation algorithm with path neighborhood information to incrementally construct a neighborhood augmented graph representation of the configuration space. As typical of wavefront propagation algorithms, the approach only utilizes local connectivity information (adjacency function) to incrementally construct and search in the neighborhood augmented graph. Path neighborhoods efficiently distinguish between the geodesic paths, which can then be used to create and maintain multiple vertices that correspond to the same configuration, but different geodesic paths leading up to it.

B. Contributions

The contributions of this paper are as follows:

- Design of a novel topo-geometric path planning approach that interfaces with any wavefront propagation algorithm and utilizes a *neighborhood augmented graph*, the construction of which is simple and does not require complex geometric constructions on the underlying configuration space or global information about its topology (Sections V-A, V-B). We also provide theoretical results on the algorithm (Section V-C).
- Demonstration of the algorithm's capabilities for finding multiple geodesic paths in 2D and 3D configuration spaces of various geometry including configuration spaces on which distinct geodesic paths belong to the same homotopy class (Section VII-A).
- An adaptation of the proposed method for computing geodesic paths in environments with extremely low curvature or cost variation where geodesic paths are almost identical (Section V-E1).

Department of Mechanical Engineering and Mechanics, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015, U.S.A., [als421,sub216]@lehigh.edu.

- Implementation of the algorithm in path planning for a tethered robot navigating in 3D and with a tether-length constraint (Section VI), along with simulations and real robot experiments in different environments (Section VII-C2, VII-C3).

II. Related Work

A. Optimal Path Planning

Graph-search based path planning has been used extensively in solving low-dimentional planning problems because of its simplicity and effectiveness [23, 24, 25, 26]. It involves a wavefront propagation approach which incrementally constructs a discrete representation of the configuration space in the form of a graph, and explores it starting from an initial configuration until a goal configuration. This is accomplished via systematic graph search algorithms such as Dijkstra's [1], A* [2] and D* [3]. In recent years, development of any-angle search algorithms [4, 5] have allowed computation of optimal paths that are not necessarily restricted to a discrete graph, and development of search algorithms for simplicial complex representations (instead of graph representations) have allowed computation of smooth paths that are optimal in the underlying configuration space [6].

Sampling-based path planning methods emerged as a scalable alternative as the motion planning problems got more complex and higher-dimensional. These methods sample configuration space points according to a scheme, check their validity through collision detection algorithms and connect them into a graph or a tree representation of the configuration space. By leveraging the sampling strategy, they produce paths without exploring as many points on the configuration space as graph-search based algorithms. However, they can only guarantee completeness and optimality in the probabilistic sense. Algorithms in this category include PRM [27], RRT [28] and their optimal variants [7].

As our approach relies on wavefront propagation to construct path neighborhoods, in this paper we focus on graph-search based path planning algorithms.

B. Topological Path Planning (TPP)

There is a wide range of topological tools that have been applied to robot motion planning problems, with the aim of computing distinct non-homotopic paths, paths constrained to a specified homotopy class, non-entangling paths for multi-robot and tethered robot motion.

In [29], braid groups are used to represent the space-time trajectories of a multi-robot system. This method allows local path deconfliction using braids and braid groups, which only capture topological properties of the joint configuration space of multiple robots and cannot reason about geodesics that may arise not only due to the topology of the underlying configuration space (e.g., due to the presence of obstacles), but also due to the underlying metric (high cost/curvature regions). The notion of persistent homology is used in [30] to classify robot trajectories, which relies on a simplicial complex representation of the configuration space constructed via a sampling-based approach. However, this approach is more concerned with the classification of the trajectories and can only provide distinct paths if large enough number of paths samples are input to the classifier, which is not a systematic way of producing distinct paths. A sampling-based approach is proposed in [31] to compute paths from diverse homotopy classes. Their method relies on the application of discrete Morse theory to a simplicial complex representation of the configuration space to identify critical points on the obstacle boundaries. However, the method is expensive as it requires a topological collapse to be performed and only allows non-homotopic paths to be computed, hence unable to compute distinct geodesic paths that are homotopic and created pure due to the underlying metric of the configuration space such as high cost/curvature regions. A k-shortest non-homotopic path planning problem is considered in [20], where the authors leverage computational geometry to identify visibility regions and construct a tree-based representation of the configuration space divided into visibility regions. Although computationally efficient, this method is restricted to motion planning problems on a 2D plane.

More relevant to our work is the use of homotopy invariants to augment the classical path planning algorithms with topological information. Each point on the configuration space, either explored on the grid or sampled randomly, is assigned a homotopy invariant, based upon the geodesic path that leads to the point. The resulting representation of the configuration space is said to be augmented, where there may exist multiple vertices representing the same configuration, if they are reached via non-homotopic paths. Then by deploying graph-search algorithms, it becomes possible to find a desired number of geodesic paths in distinct homotopy classes. For more details on TPP methods that leverage graph-search algorithms, the reader can refer to the author's prior work [15, 16, 10, 17, 18]. Homotopy-Aware RRT* algorithm by Yi et al. provides a sampling-based approach to TPP augmented with a string-based homotopy invariant, allowing paths restricted to specified homotopy classes to be found [19].

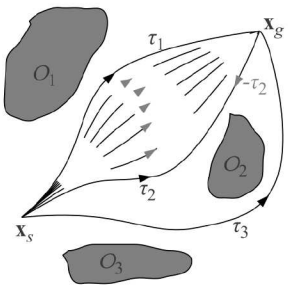
Existing work on TPP provides insight on how multiple paths can be computed when they are non-homotopic. However, it becomes challenging to apply these methods to problems where the configuration space has a more complex geometry, as the necessary constructions to evaluate homotopy invariants are difficult to generate in an automated fashion. For 2D configuration spaces with convex obstacles these representations could be as simple as some non-intersecting rays emanating from representative points – as seen in Figure 1(b). However, in higher dimensional spaces, obstacles of more complex topology and geometry may only be represented with skeletons and hyper-surfaces which are certainly more challenging to construct – as seen in Figure 1(c). In such spaces the fundamental groups are often not freely generated, and hence the computation of the homotopy invariants require complex equivalence check algorithms. Even then, algorithms discussed in this section are not able to distinguish between geodesic paths when they belong to the same homotopy class, resulting in only a subset of desired solutions being found for the k geodesic path planning problem.

C. Multi-Path Planning

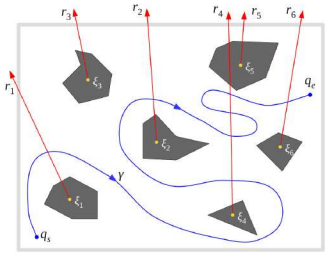
Works categorized under this section address the fact that the homotopy classes may not capture the set of all useful paths in certain configuration spaces and consider the task of distinguishing between paths even when they belong to the same homotopy class.

A heuristic method has been developed to compute diverse paths in [32]. The method works in an iterative manner, by computing the shortest path on a graph and removing edges in the vicinity of the found path in each iteration. Resulting paths are separated by a tunable distance that measures the path diversity. However, this method is concerned with finding multiple paths that are separated by the tunable threshold even when a distinction between the paths do not exist with respect to the geometry of the underlying space.

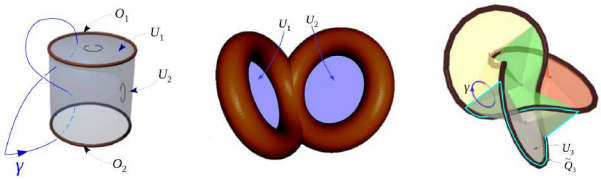
Jaillet and Simeon developed an approach for constructing probabilistic roadmaps that capture a set of paths that are difficult to deform into one another [21]. The deformation difficulty provides a finer



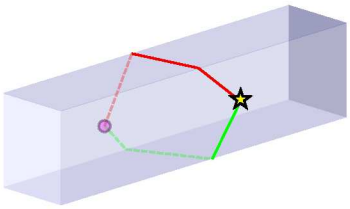
(a) Homotopy equivalences: τ_1 and τ_2 are homotopic, while τ_3 belongs to a different homotopy class.



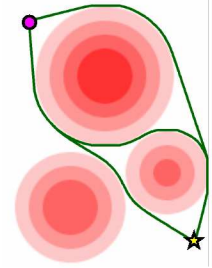
(b) In 2D environments, rays emanating from representative points on the obstacles can be used to generate word-based homotopy invariant called h-signatures [15, 16].



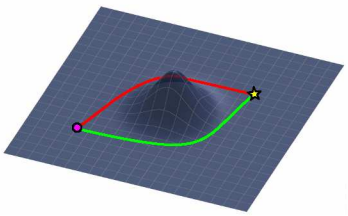
(c) In 3D environments, rays need to be replaced by surfaces bounded by obstacles with genus higher than zero. Such constructions are significantly more difficult, and the h-signature computation is highly nontrivial especially in presence of knotted or linked obstacles [18].



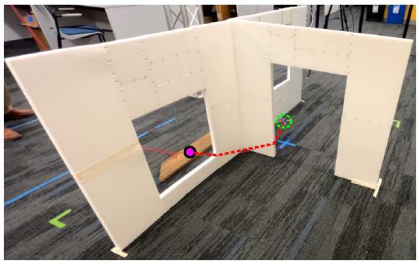
(d) There exist multiple, geodesic paths around a rectangular prism, even though these paths are topologically equivalent (belong to the same homotopy class).



(e) On a 2D environment with high cost regions, there exists multiple, geodesic paths that are topologically equivalent.



(f) On a surface with non-zero curvature, there exists multiple, geodesic paths that are topologically equivalent (can be continuously deformed into one another although it requires a path to be *stretched* over the *hill*).



(g) A tethered quadrotor performing a navigation task in a building-like environment with doors/windows need to reason about topo-geometrically distinct paths.

Fig. 1: **Background and Motivation –** *Top row* (a-c): Existing/prior work on *topological path planning (TPP)* for computing optimal paths in different homotopy classes. The constructions and computations gets significantly harder in 3-D configuration spaces, requiring complex group-theoretic reasoning in presence of obstacles that are knotted or linked (such structures are encountered in urban structures such as buildings with stairways, doors and windows). *Bottom row* (d-g): Motivation behind current work – Even in absence of topologically-distinct classes of paths, there can exists different geometrically-distinct geodesic paths (d-f). This, along with the increased complexity of TPP in 3D (c), motivates our current topo-geometric planning algorithm that creates an unified framework for computing multiple geodesic paths connecting a start and a goal configuration.

classification of paths than the notion of homotopy. An adaptation of this work to robust trajectory replanning for quadrotors is presented in [22]. As PRM based approaches, these works generate a representative roadmap of the configuration space that capture a wider variety of paths than only distinct homotopy classes. In doing so, they utilize the notion of visibility between a pair of configuration space points. To evaluate visibility between a pair of points in the configuration space, one has to construct and edge/path connecting the two configurations in the absence of obstacles and check whether this path lies in the obstacle free space [33]. The construction of these paths can be done in several ways including linear interpolation on a coordinate chart, solving the geodesic equation or an intermediate path planning problem in between the two points. The first approach makes the notion of visibility dependent on the choice of coordinate charts and thus on global information such as the topology of the configuration space and location/size of the obstacles. The latter two approaches will amount to using the solution to the problem at hand (i.e., being able to compute locally-optimal paths) to solve the problem. Additionally, in environments with high-cost regions or non-zero curvature, the presence of multiple locally-optimal/geodesic paths between two points makes the definition of visibility ambiguous. An example of such a scenario can be seen in the Figure 1(e,f), where in the absence of obstacles, using the Cartesian chart, every pair of points on the domain would be visible to each other, thus a visibility-based check deeming all paths to be equivalent even though there can exist multiple locally-optimal paths around the high-cost region. In this situation, it is even possible that one locally-optimal path may be blocked by an obstacle while another is not, making the definition of visibility ambiguous.

Orthey et al. developed an algorithm that can compute geodesic paths, categorize and present them via a tree structure [13]. This tree allows users to choose between classes of geodesic paths, with each choice leading to the exploration of a finer classification of the paths in the category. This approach also relies on visibility checks to make a distinction between paths and is subject to the same challenges.

The topo-geometric approach proposed in this paper only requires local information indicating collision-free adjacents of a configuration, in other words two configurations that are sufficiently close, which is a notion independent of the choice of the coordinate chart in most configuration spaces (configuration spaces that can be represented as manifolds with boundaries) and is defined by the existence of a small neighborhood (open set) containing those configurations. Since our configuration space is discretized as a graph, the resolution of "sufficiently close" is that of the edges in the graph. Two feasible configurations that are adjacent are, by definition, connected via an edge and they belong to a small neighborhood that locally resembles Euclidean space (tangent space of the underlying configuration

manifold). This edge can be easily represented by the straight-line constructed via linear interpolation on any coordinate chart. These properties make the algorithm proposed in this paper a local one that does not require global visibility information between distant configurations.

D. Tethered Robot Motion Planning

When the optimal path planning problem is posed for a robot with limited tether length (L), each point in the configuration space will represent a valid location of the robot and a valid configuration of its tether starting from the anchor point and ending at the robot location. This poses a challenge, as the space of all tether configurations (space of curves of length L) is infinite-dimensional. A compact representation of an elastic cable's configuration space have been developed in [34], that relies on solving the differential equations governing the equilibrium configurations of the cable. This compact representation can be leveraged by a sampling-based method to plan the necessary motions for manipulating the cable [35]. However, the cables considered in this case have large stiffness and fewer degrees of freedom than an arbitrarily flexible one, which makes it possible to represent the cable configuration space as a finite-dimensional configuration space. This allows the cable to deform around obstacles without the need of considering contact or interaction with obstacles. In contrast, the tethers we consider in this work are flexible without curvature constraints, and they can come into contact and wrap around the obstacles as the robot navigates. A more suitable modeling approach is to assume a taut tether and to consider the geodesic curves to approximate the tether configurations. In 2D environments with obstacles, this allows the configuration of a tether to be represented discretely via a word-based homotopy invariant, since there exist a single geodesic path in each homotopy class. By running a wavefront propagation algorithm for a fixed radius, Kim et al. construct a homotopy-augmented graph representing the configuration space of a tethered robot and search this graph for the optimal tethered path [15]. In their following work, authors utilize a homotopy informed heuristic together with the Multi-Heuristic A* algorithm to eliminate the preconstruction step [36].

Although homotopy based methods work effectively in 2D problems, geometrical constraints brought by the tether cannot be captured by only considering the non-homotopic paths in 3D. In 3D environments with prismatic obstacles, there may exist tether configurations that are homotopic, but when identified as the same that result in paths that violate the tether length constraint. In the literature, tethered robot planning problems in 3D are simplified either by considering projections onto 2D or assuming obstacles extending vertically to infinity [37, 38].

As the space of tether configurations is infinite dimensional and the approximate representation using word-based homotopy invariants is discrete, strategies for sampling and connecting sampled states do not work well, thus sampling-based methods are not very common. Paton et al. utilizes a sampling-based approach where the tether configurations are not sampled but predicted from the sampled robot location as a sequence of contact points with the environment [39]. However, prediction of the contact points requires computationally expensive checks for the interaction with the environment mesh and the history of contact points needs to be tracked for both sampling and connecting configurations.

Our solution to the tethered robot motion planning problem based on the neighborhood-augmented graphs captures the necessary geometrical properties, avoids any violation of the tether length constraint and does not require any simplification to the actual environment.

III. Preliminaries

In this section, we provide background on homotopy classes of paths, geodesic paths and augmented graph search based planning to aid the discussion on the k geodesic path planning problem and the proposed solution using neigborhood-augmented graphs.

A. Path Properties

To better explain the difference between k geodesic path planning and k non-homotopic path planning problems, we provide a definition of the homotopy first:

Definition 1 (Homotopy Classes of Paths)**.** *Two paths connecting the same start and goal points in a configuration space are said to be in the same* homotopy class *(or* homotopic*) if one can be continuously deformed into another without intersecting/crossing obstacles (Figure 1(a)). Otherwise they are called* non-homotopic*.*

Homotopy classes are the main topological classes of interest when it comes to paths or trajectories in a configuration space. One common way to classify the paths and keep track of their classes is to use a *homotopy invariant* as discussed in Section II-B.

In this work, we consider the robot configuration spaces on which paths, infinitesimal perturbations of paths, and small neighborhoods of configuration points are well defined. These conditions are satisfied by Riemannian manifolds with boundaries, which constitute the types of configuration spaces that we will consider. We formalize the notion of *geodesic paths* as follows:

Definition 2 (Geodesic Paths [40])**.** *A path connecting a fixed pair of start and goal points is called* geodesic *if any infinitesimal perturbation to the path results in an increase in the cost of the path.*

An immediate observation is that given a start and a goal point in a configuration space, there can be multiple distinct geodesics connecting them that may or may not be in different homotopy classes (Figure 1(d-f)). The presence of distinct homotopy classes give rise to distinct geodesic paths (at least one in each homotopy class) [41]. But even within the same homotopy class there can be multiple geodesic paths created due to geometry, curvature and non-uniform cost.

B. Augmented Graph Search Based Planning

1) Discrete Graph Representation of a Configuration Space for Optimal Path Planning: During graph-search based optimal path-planning, a configuration space graph $G=(V,E)$, is constructed as a discrete representation of the configuration space, where the vertex set (V) consist of points from the free configuration space and the edge set (E) contains the edges connecting adjacent vertices.

Search-based algorithms such as Dijkstra's, S*, construct the graph via a wavefront propagation approach: Starting from a start vertex, $q_s \in V$, they sequentially explore adjacent vertices, while maintaining a priority queue of vertices sorted by the g-score (the optimal cost-to-come from the start vertex q_s). Expanded vertices are popped from the front of the queue, whereas newly generated vertices are inserted based on the g-scores. We refer to the maintained priority queue as the *open list* or the *exploration front* and the set of expanded vertices as the *closed list*. For a detailed discussion on graph-search please refer to [42]. This expansion process continues until a desired goal vertex, $q_g \in V$, is reached. An optimal path from q_s to q_g can be constructed by following the steepest decrease in the g-scores, a procedure referred to as *path reconstruction*.

At the core of this process, algorithms require the following modules, which implicitly represents the structure of G:

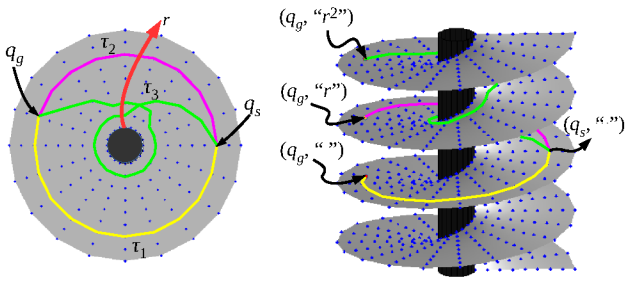


Fig. 2: Visualization of the h-augmented graph [15, 16, 43]: *Left:* A planar configuration space (light gray), with a single obstacle (black), and the vertex set, V (blue dots), in it. *Right:* The vertex set, V_h, in the h-augmented graph. Note how the paths *lift* to have different goal points of the form $(q_g,*) \in V_h$ corresponding to the same goal $q_g \in V$.

i. An *adjacency function*, $\mathcal{A}$, that returns the valid *adjacent vertices* of the vertex $q \in V$ (that will be connected to q by edges)
ii. A *cost function*, $\mathcal{C}$, that returns the cost of an edge
iii. A *vertex equivalency function*, $\mathcal{Q}$, to check whether two vertices are identical. This module is required to determine if a vertex has been previously visited (already inserted in V). For the usual discrete graph representation of a configuration space, this is usually as simple as comparing the coordinates of the new vertex with the coordinates of the already expanded vertices. Note that for configuration spaces such as $\mathbb{S}^1$, this function implements the identification of points 0 and 2π.

Without any other way to distinguish between vertices besides their coordinates, graph-search based methods can only maintain the globally optimal path leading to a configuration. Augmented graphs allow us to keep track of multiple geodesic paths.

2) Augmented Graph for Multi-Path Planning: The graph representation of the configuration space can be augmented with additional information based on path properties to make a distinction between vertices that are reached via different paths. The resulting representation is an *augmented graph*, $G_A = (V_A, E_A)$. The vertices in V_A are of the form $v = (q, \alpha)$ for $q \in V$ denoting the configuration and α the path property (e.g. homotopy class, path neighborhood). In the case of an augmented graph, $\mathcal{Q}_A$ compares both the configuration and the the path property. Thus, corresponding to a vertex $q \in V$ there may exist multiple vertices of the form $(q,\alpha_1),(q,\alpha_2),\cdots \in V_A$, each corresponding to the configuration q reached via a different path. The edges and associated costs are determined by $\mathcal{A}_A, \mathcal{C}_A$, which can be designed based on $\mathcal{A}, \mathcal{C}$. The search for k paths on an augmented graph terminates once $(q_g,\alpha_1),\ldots,(q_g,\alpha_k)$ are reached. k distinct optimal paths can be reconstructed for each $(q_g,\alpha_1),\ldots,(q_g,\alpha_k)$.

The idea of augmented graph is most easily demonstrated by the construction of h-augmented graph in context of topological path planning in multiple homotopy classes [15, 16, 43]. Figure 2 provides a visualization of the h-augmented graph.

IV. Problem Definition and Analysis

In this section, we define the problem of finding k geodesic paths and relate it to the topo-geometrically distinct paths via the notion of path neighborhoods.

A. Problem Definition

Definition 3 (k geodesic path planning problem)**.** *Given a configuration space equipped with a path metric, a start and a goal configuration, the k geodesic path planning problem seeks to find k geodesic paths between the start and goal configuration in the free configuration space.*

The purpose of this paper is to develop an algorithm to solve the k geodesic path planning problem.

B. Problem Analysis using Path Neighborhoods

We investigate the distinctive properties of geodesic paths to develop a solution using a wavefront propagation type algorithm.

Remark 1 (Key Observation about Tangents to Geodesics). *Two distinct geodesic paths between a start and a goal point intersect at least at one point q on which the tangents to both paths are distinct, causing paths to diverge from each other (Figure 3).*

Remark 2 (Path Neighborhoods as Proxy for Tangent Vectors). *The tangent space of the configuration space at a point is reasonably represented by a small neighborhood around the point. Therefore, the tangent vector to a path at a point can be approximately represented by a neighborhood around the path at that point (see Figure 3). We call such a representative neighborhood a* path neighborhood.

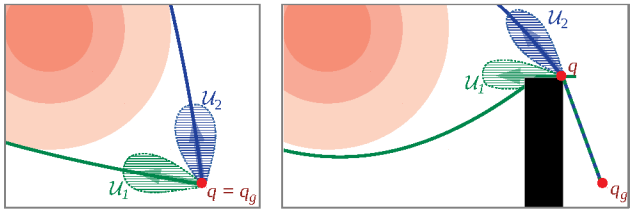


Fig. 3: Optimal/geodesic paths to a point q_g in the configuration space that are topo-geometrically distinct have distinct tangent vectors at a point q (with $q = q_g$ in case (a), but a point other than q_g in case (b)). Instead of comparing and distinguishing the paths by their tangent vectors (which is often computationally difficult) at a point where the paths meet, a reasonable approach is to compare the *path neighborhoods* (shown by hatched regions) in the proximity of the paths near the point q.

Motivated by these observations we provide the following definition:

Definition 4 (Topo-geometrically Distinct Paths)**.** *Two distinct geodesic paths connecting the same start and target points, q_s and q_g, are called* topo-geometrically distinct *if there exists a common point, q, on the paths at which they have distinct path neighborhoods.*

In a wave-front propagation type algorithm[1], starting from q_s, this definition is particularly relevant for identifying topo-geometrically distinct paths to the point where different *branches* of the wave-front meet for the first time around an obstacle and/or high-curvature/cost regions (see Figure 4(a-c)). Such a point lies on the *cut locus* of q_s [44], and identification of topo-geometrically distinct paths to it based on distinct path neighborhoods (Figure 4(c)) is critical in being able to identify, create and maintain distinct branches of the wave-front. Once identified, the branches can be kept separated by distinguishing between vertices that represent the same configuration reached via different branches (Figure 4(d-f)). Subsequently, configurations

[1] In practice, this is Dijkstra's or S* algorithm on a discrete graph representation of the configuration space as will be discussed in the next section

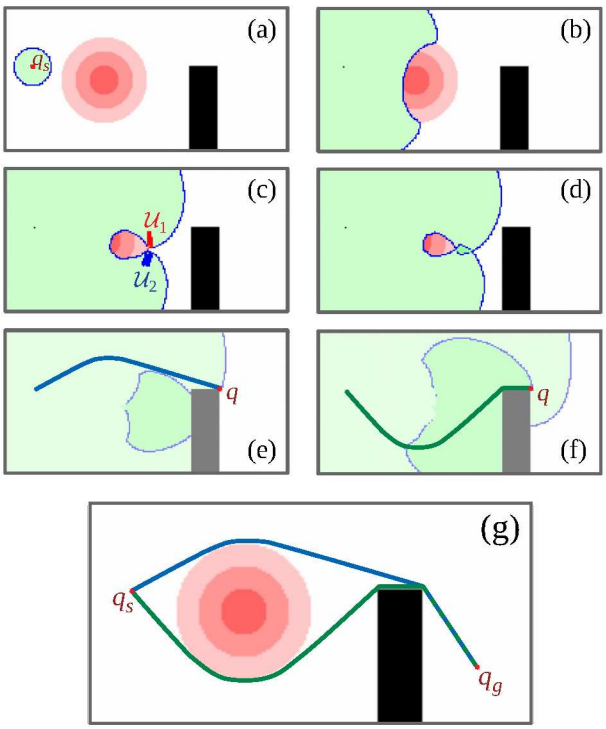


Fig. 4: Overview of the proposed *topo-geometric path planning* using search in *neighborhood-augmented graph*: (a-b) Search propagates as a uniform wave-front before encountering any obstacle or high-cost region. (b-c) Two branches of the search emerge as the search wave-front is disturbed by the increased costs. (c) When two branches meet, paths leading up to the same configuration point will have disjoint path neighborhood sets ($\mathcal{U}_1$ and $\mathcal{U}_2$). (d) Branches stay separate as a result. Two topo-geometrically distinct paths to q are found as (e) the first branch reaches the point q, (f) followed by the second branch. (g) Eventually the branches reach q_g and two topo-geometrically distinct paths are obtained. The figures are actual **results** from the proposed algorithm.

reached via the topo-geometrically distinct branches have different path neighborhoods and hence can be identified as distinct with topo-geometrically distinct paths leading to them (see Figure 4(e-g)).

At this point we do not explicitly define or restrict the shape or size of the *path neighborhoods*. The effect of parameters determining the shape and size of these neighborhoods will be discussed later along with ways of estimating these parameters for a given configuration space. Under some conditions regarding the curvature of the configuration space, all geodesic paths are topo-geometrically distinct. As a result, an algorithm capable of computing k topo-geometrically distinct and geodesic paths solves the k geodesic path planning problem.

V. Algorithm Development

A. Neighborhood-augmented Graph

In order to do away with complex geometric constructions in topological path planning, and to be able to find geodesic paths that are topo-geometrically distinct (Definition 4), in this paper we propose a *neighborhood-augmented graph* (NAG), $G_N=(V_N,E_N)$, that is constructed based on the given discrete graph representation of the underlying configuration space, $G=(V,E)$, by augmenting each vertex with a "*path neighborhood*" that is representative of the tangent vector of the path leading to the vertex (Figure 3). Just like the previously-described augmented graphs, corresponding to a vertex $q \in V$, there can exist multiple distinct vertices, $(q,\mathcal{U}_1),(q,\mathcal{U}_2),\cdots \in V_N$, where each $\mathcal{U}_i$ is a set consisting of **pointers**[2] **to** vertices in V_N (*i.e.*, $\mathcal{U}_i \subset V_N$) that make up the neighborhood of the path leading to the vertex $(q,\mathcal{U}_i)$. We refer to each such $\mathcal{U}_i$ as a *path neighborhood set* corresponding to the path leading to the vertex v.

For topo-geometrically distinct paths, the path neighborhood sets are expected to be disjoint as illustrated in Figure 3. While, for a configuration reached via paths that are topo-geometrically similar, there will be significant overlap between the path neighborhood sets of the corresponding vertices. Thus, we define

Definition 5 (Equivalence of Vertices in G_N)**.** *Given $(q,\mathcal{U}),(q',\mathcal{U}') \in V_N$, they are called* equivalent *(i.e., considered to be the same vertex, and denoted as $(q,\mathcal{U}) \equiv (q',\mathcal{U}')$), if $q \equiv q'$ and $\mathcal{U} \cap \mathcal{U}' \neq \emptyset$.*

This definition of the vertex equivalency determines the behavior of $\mathcal{Q}_N$ (as described in Section III-B) and is critical in incremental construction of the graph, G_N – every time a vertex is generated, it can be compared against the vertices in the closed list to determine whether it is a new vertex or identical to an existing one.

An Implementational Detail: It is to be noted that for two path neighborhood sets to intersect (*i.e.*, for the computation of $\mathcal{U} \cap \mathcal{U}'$), they must contain a common vertex that not only has the same coordinates/configuration but also the same path neighborhood sets (that is, $\mathcal{U} \cap \mathcal{U}' \neq \emptyset \iff \exists\, (w,\mathcal{W}) \in \mathcal{U}$, $(w',\mathcal{W}') \in \mathcal{U}'$ s.t. $(w,\mathcal{W}) \equiv (w',\mathcal{W}')$). At a first glance it may appear that the computation of equivalence thus requires an iterative computation of other equivalences for vertices in the path neighborhood sets. However, in implementation, since path neighborhood sets are stored as *pointers* to vertices in V_N, it is sufficient to check if the sets of pointers, $\mathcal{U}$ and $\mathcal{U}'$, have any common pointer. If two pointers are identical, that will automatically imply that the corresponding vertices are identical (since otherwise they would have been identified as different vertices in the first place and would have been assigned different pointers).

Adjacency Function: For any vertex $(q,\mathcal{U}) \in V_N$, we can utilize the adjacency function of the configuration space graph, G, which returns the configuration of vertices that are adjacent to q. Suppose $\mathcal{A}(q)=\{q_1,q_2,\cdots\}$. The path neighborhood sets of each q_i is then constructed by computing a neighborhood of $(q,\mathcal{U})$ via a short secondary search in the current G_N. The details of this secondary search is described in Section V-B. If this secondary search returns a path neighborhood set $\mathcal{U}'$, the adjacency function of G_N can then be described as $\mathcal{A}_N((q,\mathcal{U}))=\{(q_1,\mathcal{U}'),(q_2,\mathcal{U}'),\cdots\}$. The cost of each edge on G_N is the same as its projection on the original graph G (that is, $\mathcal{C}_N((q,\mathcal{U}),(q_i,\mathcal{U}'))=\mathcal{C}(q,q')$).

The construction of the graph G_N can be performed using any graph-search algorithm, given the implicit representation of G via $\mathcal{A},\mathcal{C}$ and $\mathcal{Q}$. Throughout this paper we have presented results obtained using Dijkstra's and S* [6] algorithms for computing different topo-geometrically distinct paths that are optimal in the underlying configuration space.

As a simple illustration, however, we present a pseudo-code for the approach using Dijkstra's algorithm that incrementally constructs a neighborhood-augmented graph in Algorithm 1. The traditional way to use this algorithm is to insert the goal-based stopping criteria given

[2]A pointer refers to the address/reference in the computer memory where the data associated with a vertex is stored.

Algorithm 1 Incremental Construction and Dijkstra's Search in a Neighborhood-augmented Graph (NAG)

$G_N = searchNAG\ (q_s, \mathcal{A}, \mathcal{C}, @stopSearch)$

Inputs:
a. Start configuration $q_s \in V$
b. Adjacency function $\mathcal{A}$ (describing the connectivity of graph G)
c. Cost function $\mathcal{C}: V \times V \to \mathbb{R}^+$
d. Stopping criteria (function), *stopSearch*: $V_N \to \{0,1\}$

Output:
Graph G_N, with g-scores and neighborhoods computed for every vertex

1: $v_s := (q_s, \{\&v_s\})$ // start vertex in V_N, with self-reference // in its path neighborhood set.
2: Set $g(v_s) = 0$ // g-score
3: $V_N = \{v_s\}$ // vertex set
4: $E_N = \emptyset$ // edge set, maintained implicitly as a link tree/graph
5: $Q = \{v_s\}$ // open list, maintained by a heap data structure.
6: $v := v_s$
7: **while** $Q \neq \emptyset$ **AND not** *stopSearch*(v) **do**
8: $v := (q, \mathcal{U}) = \text{argmin}_{v' \in Q} g(v')$ // heap pop.
9: $Q = Q - v$ // heap pop.
10: $\mathcal{U}' = computePNS\,(v, G_N)$ // path neighborhood set // for path leading to v
11: **for** $q' \in \mathcal{A}(q)$ **do**
12: $v' := (q', \mathcal{U}')$ // potential adjacent vertex
13: $g' = g(v) + \mathcal{C}(q, q')$ // potential g-score for v'
14: **if** $\nexists\, w \in V_N$, with $v' \equiv w$ **then** // new vertex
15: $V_N = V_N \cup \{v'\}$
16: $E_N = E_N \cup \{(v, v')\}$ // maintained as linktree // for adjacency function computation, $\mathcal{A}_N$
17: $g(v') = g'$
18: $Q = Q \cup \{v'\}$
19: v'.came_from $= v$
20: **else** // vertex already exists (w)
21: $E_N = E_N \cup \{(v, w)\}$
22: **if** $g' < g(w)$ **AND** $w \in Q$ **then** // update w
23: $g(w) = g'$
24: w.came_from $= v$
25: $w.\mathcal{U} = \mathcal{U}'$
26: $G_N := (V_N, E_N)$
27: **return** G_N

Algorithm 2 Goal-Based Stopping Criteria for Search

[bool, $\mathcal{V}_g$] = $stopSearch\_AtGoal_{q_g,k}(v)$

Input: Current vertex $v = (q, \mathcal{U}) \in V_N$
Static variables / parameters:
a. Goal configuration $q_g \in V$
b. Number of paths to find k

Output:
a. Boolean (**true** to stop search, **false** to continue)
b. Vertices in the NAG found at goal configuration $\mathcal{V}_g$

1: **static** $\mathcal{V}_g (= \emptyset)$ // static variable initiated with empty set
2: **if** $v.q = q_g$ **AND** $v \notin \mathcal{V}_g$ **then**
3: $\mathcal{V}_g = \mathcal{V}_g \cup v$
4: **return** $(|\mathcal{V}_g| == k)$

in Algorithm 2 as the input, provided a goal coordinate and number of desired paths. However, depending on the application, there could be other uses of the algorithm, utilizing alternative stopping criteria, as it will be further discussed in Section VI-B.

Figure 5 illustrates the incremental construction of a neighborhood-augmented graph based on a given discrete graph. Two separate search *branches* emerge due to a hole in the underlying discrete graph. Neighborhoods for selected vertices are shown in the bottom row. Vertices shown in the first two columns are trivially distinct as they do not share the same coordinates. In the third column, two vertices with the same coordinates (*i.e.*, the same vertex in G) are selected for which the path neighborhood sets are disjoint as they are reached via different branches of the search. As a result, two distinct vertices (with common coordinates but with two disjoint path neighborhood sets) are maintained within the neighborhood-augmented graph. Whenever a distinct vertex at the goal coordinates is expanded, it is possible to reconstruct a topo-geometrically distinct path from the start to the goal coordinate using the corresponding goal vertex and the constructed neighborhood-augmented graph with a *path reconstruction* subroutine. Two of such paths are shown in the last column of Figure 5.

For given shape characteristics/geometry of the path neighborhood sets, and the geometry of the artifacts on the configuration space influenced by holes, obstacles, high-cost or non-zero curvature regions, or topology, it thus is possible that vertices sharing the same coordinates, but reached via topo-geometrically distinct paths, have non-intersecting path neighborhood sets and hence are inequivalent in the neighborhood-augmented graph (Figure 5(c)). This creates distinct branches of the search around such artifacts that stay separate (cross-over) and progress further. The effect of the size/geometry of the path neighborhood sets on the ability to find topo-geometrically distinct paths around artifacts of different size/geometry is discussed in more detail in Section V-D. Performing the search on the neighborhood-augmented graph computes geodesic paths, in addition to the globally optimal path, that are topo-geometrically different.

B. Neighborhood Generation

During the construction of the neighborhood-augmented graph (Algorithm 1 – also referred to as ***primary search*** henceforth), a path neighborhood set, $\mathcal{U}$, of a vertex $v \in V_N$ needs to be computed. The path neighborhood set consists of a set of vertices within some distance r_n of v in the neighborhood of the path leading to v in the neighborhood-augmented graph – Line 10 of Algorithm 1. To compute this path neighborhood set, a *secondary search* is performed on the current neighborhood-augmented graph, G_N (using A* algorithm), starting from the vertex of interest v. We emphasize that this secondary search (which will be referred to as ***neighborhood search*** from now on) occurs on the existing graph, G_N, during which G_N remains unchanged. A pseudo-code for the neighborhood search (*computePNS* routine) is provided in Algorithm 3.

If the neighborhood search is carried out until a distance r_n from v (reaching a g-score of r_n), the expanded vertices will form a shape similar to a disk-section around v, bounded by a circle of radius r_n at the upstream and by the open list of the primary search at the down-stream (Figure 6(a)). However, a disk-shaped set does not provide a reasonable representation for a neighborhood around the path leading up to v. By using the g-score from the primary search as heuristic

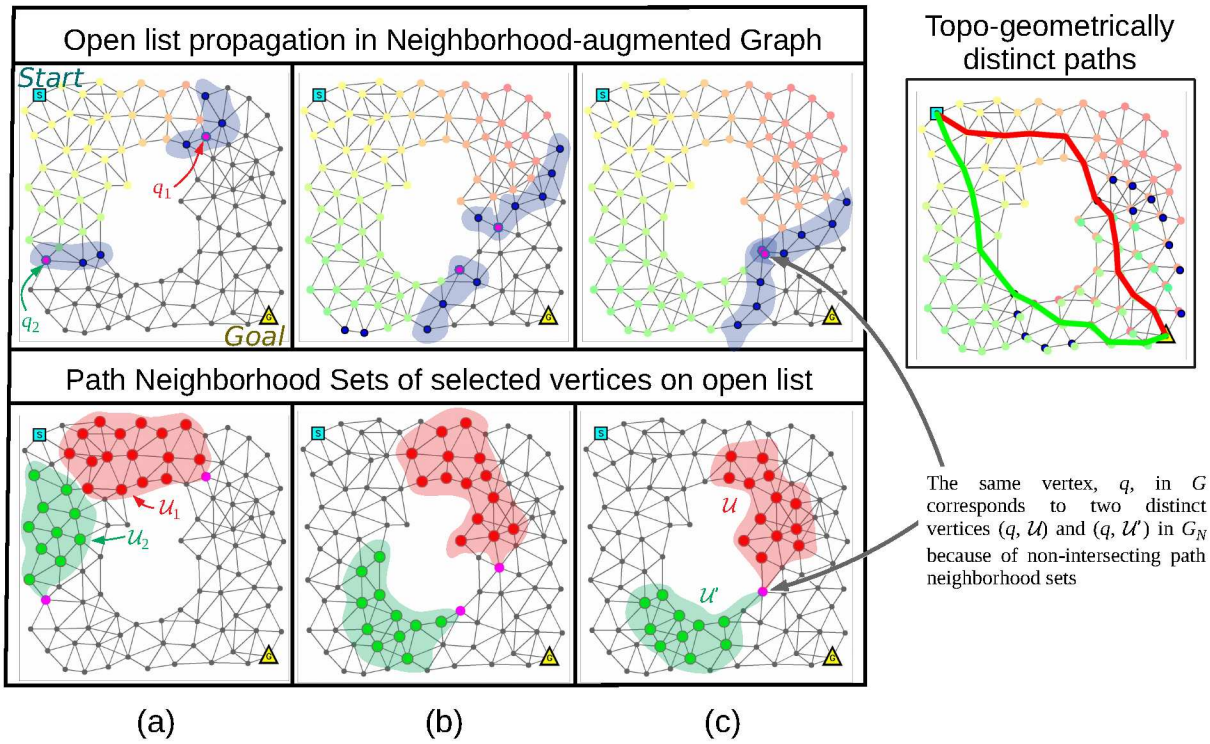


Fig. 5: Path planning on neighborhood-augmented graph, G_N, using Dijkstra's search (actual **result**, with highlights and labeling for aiding explanation). The planar graph that is visible in the figures is the configuration graph $G=(V,E)$. The neighborhood-augmented graph is constructed incrementally, and the top row shows the vertices in the *open list* at different instants during the progress of the search (colored/highlighted in blue). However, a vertex in the neighborhood-augmented graph, G_N, is not just a vertex in G, but also contains information about a path neighborhood set leading to the vertex (maintained as a list of pointers to already-generated vertices in V_N). This is shown in the second row, where two vertices, $(q_1,\mathcal{U}_1)$ and $(q_2,\mathcal{U}_2)$, in the two different *branches* of the search (created due to the central *hole*) in the graph, are chosen to illustrate their respective path neighborhood sets (shown/highlighted in red and green respectively). However, as demonstrated in column (c) of the figure, when the two branches *meet*, corresponding to the same vertex q in the configuration space graph, we end up constructing two distinct vertices, $(q,\mathcal{U})$ and $(q,\mathcal{U}')$, in G_N. Two topo-geometrically distinct geodesic paths to the goal are found as a result (last column). The cost function used for this search example is the Euclidean distance between the vertices.

function to guide the secondary search, we obtain a path neighborhood set that *hugs* the path more closely. In practice, we use a parameter, ω (called a *heuristic weight*), to determine the influence of the said heuristic function (Line 16 of Algorithm 3) – a heuristic weight of 0 will create a disk-shaped neighborhood, while positive and larger heuristic weights will create more path-hugging neighborhood sets. We choose a $\omega \in [0,1]$. Figure 6(a,b) illustrates a comparison between a disc-shaped neighborhood and a path-hugging neighborhood.

A path neighborhood set of a vertex needs to satisfy following conditions to achieve desired branching behavior around the artifacts of the configuration space:

1) Vertices at the same coordinates generated through parent vertices that are close to each other in the graph should have large overlap in path neighborhood set. Essentially, a wavefront should not be giving rise to any branches unless it encounters an artifact in the configuration space. Size of the path neighborhood sets of nearby vertices in the same branch of a search should be large enough to contain common vertices. An example of this scenario is illustrated in Figure 7, where a vertex v, is generated by two different parent vertices v_{p1} and v_{p2} on the same branch whose path neighborhood sets have a large intersection. As a result, only a single vertex will be created and maintained at that point without creating any spurious branches.
2) For an artifact to generate multiple topo-geometric branches during the search, the size of path neighborhood set should be smaller than the size of the artifact to ensure that there will not be any intersection at the upstream of the artifact (Figure 6(c,d)). Hence, vertices generated via parent vertices on a different branch will be identified as different.

The effect of the size of path neighborhood sets relative to the obstacle size can be observed in Figure 6(c,d). As observed empirically, disk-shaped path neighborhood sets are more sensitive to the size parameter than the *path-hugging* path neighborhood sets as they tend to overlap at the upstream of an artifact even for smaller radii. Therefore, *path-hugging* path neighborhood sets are to be considered as the default path neighborhood sets shape unless stated otherwise.

C. Theoretical Analysis

1) A Sufficient Condition for Creating Topo-geometrically Distinct Branches: We formalize the conditions for being able to compute topo-geometrically distinct branches at a configuration q. Let $r_s \in \mathbb{R}^+$ be the g-score at the open list at an instant of the search under the assumption that vertices on the open list have same/similar g-scores at the same instant. We consider the closed list generated during

Algorithm 3 Path Neighborhood Set (PNS) Computation using A* Search

$\mathcal{U} = computePNS_{\{r_n,\omega,r_b\}}(v_p, G_N)$

Inputs:

a. Existing/current neighborhood-augmented graph $G_N = (V_N, E_N)$, along with the g-scores of vertices in the graph, $g(\cdot)$.
b. Parent vertex $v_p = (q_p, \mathcal{U}_p) \in V_N$

Geometry Parameters:

a. Neighborhood radius, $r_n \in \mathbb{R}^+$
b. Heuristic weight, $\omega \in [0,1]$
c. Rollback amount, $r_b \in \mathbb{Z}_{\geq 0}$

Output:

A set of vertices (*path neighborhood set*), $\mathcal{U} \subset V_N$

$v_s := rollback(v_p, r_b)$ // start vertex for secondary search
$\mathcal{U} := \emptyset$
$\widetilde{g}(v) = \infty, \widetilde{f}(v) = \infty, \forall v \in V_N$ // implicitly set all f- and g- score for secondary search to infinity for all vertices
$\widetilde{g}(v_s) := 0$ // g-score in secondary search
$\widetilde{f}(v_s) := \widetilde{g}(v_s) + \omega g(v_s)$, // f-score in secondary search. g refers to the g-score from the primary search, which is being used as a heuristic function for the secondary search
$\mathcal{U} = \{v_s\}$ // vertex set for neighborhood search
$\widetilde{Q} := \{v_s\}$ // open list for neighborhood search
$v := v_s$
while $Q \neq \emptyset$ **AND not** $\widetilde{g}(v) > r_n$ **do**
  $v := \text{argmin}_{v' \in \widetilde{Q}} \widetilde{f}(v')$
  $\widetilde{Q} = \widetilde{Q} - v$
  for $v' \in \mathcal{A}_N(v)$ **do**
    $\bar{g}' = \widetilde{g}(v) + \mathcal{C}_N(v, v')$
    if $\bar{g}' < \widetilde{g}(v')$ **then** // better g-score
      $\widetilde{g}(v') = \bar{g}'$
      $\widetilde{f}(v') = \bar{g}' + \omega g(v')$
      $\widetilde{Q} = \widetilde{Q} \cup \{v'\}$
      $\mathcal{U} = \mathcal{U} \cup \{v'\}$
return $\mathcal{U}$

the wavefront propagation (which is a discrete and neighborhood augmented representation of the explored subset of the configuration space) to be a metric space with metric/distance function, d, so that for any $p_1, p_2 \in V_N$, $d(p_1, p_2)$ is the total distance/cost of the shortest path through the set of already-explored vertices connecting p_1 and p_2. Suppose a configuration $q \in V$ is being reached via two distinct geodsics/shortest paths (*i.e.*, being generated from two different parents), γ_1 and γ_2 as illustrated in Figure 8.

We try to understand the condition under which the path neighborhoods of these two different paths, $\mathcal{U}_1$ and $\mathcal{U}_2$, will not overlap, hence giving rise to two distinct vertices $x_1 = (q, \mathcal{U}_1)$ and $x_2 = (q, \mathcal{U}_2)$, in the neighborhood augmented graph. Let γ_i' be the segment of the path, γ_i, contained within $\mathcal{U}_i$, $i = 1,2$. Then the following result holds:

Proposition 1.

If $sep(\gamma_1', \gamma_2') > \frac{4r_n}{1-\omega}$, *then* $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$.

where, $sep(\gamma_1', \gamma_2') = \inf_{v_1 \in \gamma_1', v_2 \in \gamma_2'} d(v_1, v_2)$ *is the separation between the path segments* γ_1' *and* γ_2'.

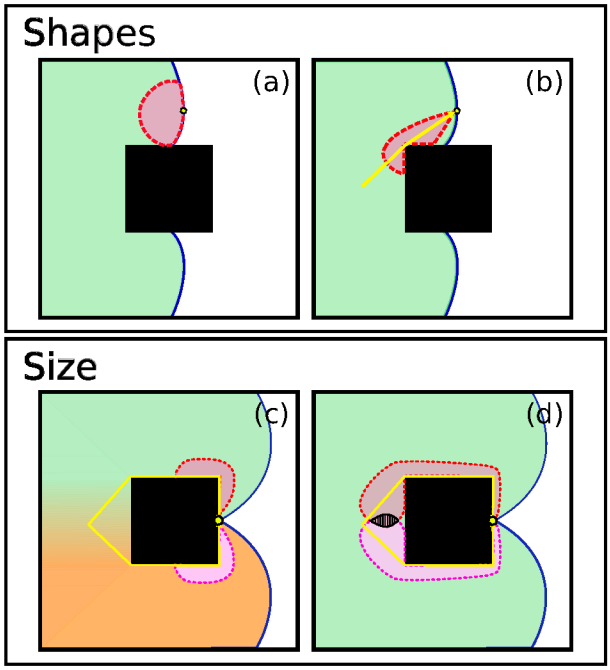


Fig. 6: (a) Disk-shaped neighborhood at a vertex of interest (VOI). (b) *Path-hugging* neighborhood at a VOI (Default option for path neighborhood sets). (c) Disjoint neighborhoods for two vertices at the same coordinates due to the obstacle resulting in the second copy to be inserted to the neighborhood augmented graph. (d) Intersecting neighborhoods at the upstream of the same obstacle, when the neighborhood size is increased. The second vertex is equivalent to the first one, thus it will be discarded.

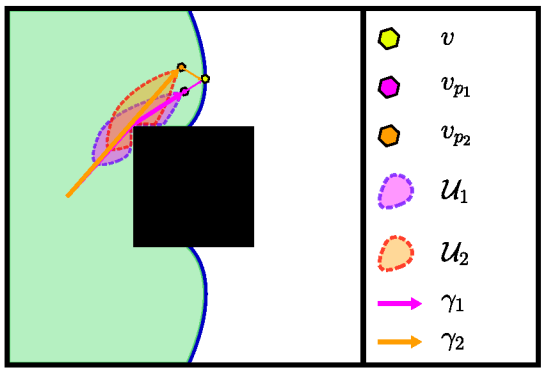


Fig. 7: A vertex at $v.q$ is generated by two distinct parent vertices v_{p1} and v_{p2}. Shortest paths γ_1 and γ_2 leading up to the parent vertices are also distinct. However, the path neighborhood sets U_1 and U_2 intersect as the paths are only slightly different. In this case, there is no need to construct and maintain another vertex at the coordinates $v.q$.

The proposition, in effect, gives a value for the minimum separation between two equal-length geodesic paths leading to the same vertex that will result in identification of the vertices x_1 and x_2 to be topo-geometrically distinct, and hence will give rise to two distinct branches in the neighborhood-augmented graph.

Here, we highlight that the separation between path segments is not an intrinsic geometrical property of the underlying configuration space, as it depends on the specific paths. Therefore, even though an insightful theoretical result involving both r_n and ω, Proposition 1 does not provide a direct way of determining a working interval for the

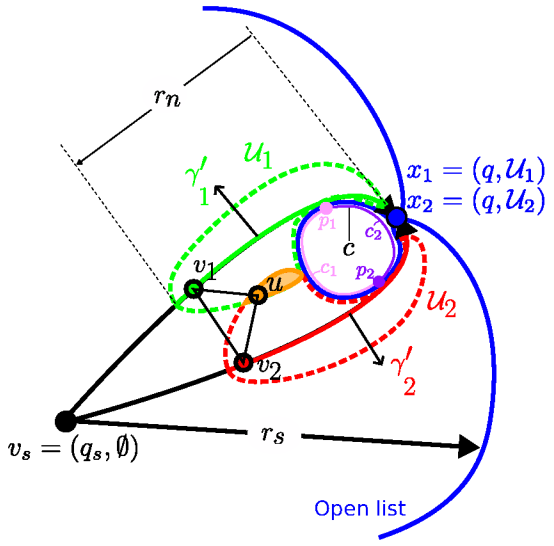


Fig. 8: Path neighborhood sets $\mathcal{U}_1$ and $\mathcal{U}_2$ are shown for two paths γ_1,γ_2 , leading to vertices x_1 and x_2 at configuration q in the open list. Neighborhood radius is r_n and the radius at the open list is r_s. v_1 and v_2 are points on attached path segments γ'_1 and γ'_2 . u is a common neighbor, such that $u \in \mathcal{U}_1 \cap \mathcal{U}_2$. γ_1,γ_2 are tangent to the closed geodesic curve c at p_1,p_2 respectively, dividing it into two segments c_1 and c_2.

parameters based purely on the properties of the given configuration space. Hence we propose an alternative but more conservative result, which allows us to choose a suitable r_n based on the intrinsic properties of the configuration space. More specifically, we use the notion of closed geodesics defined as below:

Definition 6 (Closed Geodesics)**.** *A geodesic starting and ending at the same point in a path metric space is called a closed geodesic.*

Examples of closed geodesics include great circles on a sphere and tightest paths enclosing obstacles on a configuration space. Let c^* be the shortest closed geodesic created around an artifact such as an obstacle, a high cost region, or a topological hole in the underlying configuration space, and let $l(c^*)$ denote the length of c^*. Then, we can use the following proposition to relate r_n to the length of the shortest closed geodesics, and hence be able to choose an appropriate value of r_n:

Proposition 2. *Picking $r_n < \frac{l(c^*)}{2}$ ensures that $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$, for a pair of vertices $x_1 = (q, \mathcal{U}_1)$,$x_2 = (q, \mathcal{U}_2)$ generated at the cut locus of q_s via topo-geometrically distinct paths.*

Note that Proposition 2 only requires information about the length of the shortest closed geodesic and not the computation of the geodesic itself. We provide a discussion on the estimation of closed geodesic lengths in Section VII-A6.

The proofs for the propositions presented in this section are deferred to the Appendix.

2) Complexity of the Neighborhood Augmented Graph Search: The Algorithm 1 is essentially a search in the neighborhood-augmented graph (in particular, it's illustrated with the Dijkstra's search algorithm, although any graph search algorithm can be used). The main difference is the computation of the path neighborhood set every time a vertex in the neighborhood-augmented graph is generated. The construction of the path neighborhood set of each vertex involves running a separate A* search up to a fixed distance of r_n in the partially-constructed neighborhood-augmented graph, and hence by itself is a constant-time process on an average (while the path neighborhood sets of different vertices may contain different number of vertices depending on the cost in the neighborhood, in a uniform-cost region this number is almost constant, and on a high cost region this number is lower). Thus, the complexity of the search in the neighborhood-augmented graph is same as the complexity of the search algorithm used, only with a constant multiple. For example, the complexity of the Dijkstra's search in a graph with an almost-uniform vertex degree (e.g., graph constructed from uniform discretization of a configuration space and by connecting each vertex with its neighbors) is $O(|\mathcal{V}|\log(|\mathcal{V}|))$, where $|\mathcal{V}|$ is the number of vertices expanded. The complexity of searching in the neighborhood-augmented graph using Dijkstra's search is thus also $O(k_n|\mathcal{V}|\log(|\mathcal{V}|))$, where the constant k_n is the constant-time complexity of generating the path neighborhood of each vertex. However, note that the size of the search graph, and hence $|\mathcal{V}|$, grows exponentially with the dimensionality of the underlying configuration space.

The proposed topo-geometric path planning algorithm inherits the completeness and optimality guarantees of the graph search algorithm being used for the primary search to find the k most optimal paths, provided there does not exist pathological situations created by extremely low cost/curvature regions as will be described in Section V-E.

D. Implementation Details

The main ideas and procedures of the search on a neighborhood augmented graph is as described in previous sections and the pseudocodes within. In this section, we will describe some implementational details that are not captured in previous sections.

Rollback to grandparent. In previous sections, it is assumed that the neighborhood search for a vertex v' starts from its parent vertex v. As a result, path neighborhood sets are always illustrated as attached to the vertex. However, depending on the primary search algorithm being used, at the time of neighborhood search, vertex v might not be inserted in the neighborhood-augmented graph, thus making it impossible to perform a search starting from v. This mainly occurs when using S* search algorithm, which generates the grandchildren of a vertex when expanding a vertex. A solution is to *rollback* from the vertex v' for some fixed number of generations (which means to consider the parent or n-th grandparent of the vertex) and start the neighborhood search from that vertex. This procedure is performed by the *rollback* subroutine in Line 1 of Algorithm 3, which recursively calls the 'came_from' attribute of a vertex for desired number of times. A rollback of $r_b = 3$ is found to be suitable for the algorithms we use in this paper. Note that the amount of rollback needs to be accounted for when using Proposition 2, which usually requires us to pick an even smaller r_n.

Bound on neighborhood search depth. In previous sections, neighborhood radius r_n was expressed in terms of the g-scores on the corresponding graph. However, in an environment that is uniformly discretized, on a high-cost region, a neighborhood whose radius is based upon the g-scores only may contain very few vertices. To alleviate this situation, we place a lower bound on the depth in terms of number of generations from the start vertex of the neighborhood search that must be performed for a the neighborhood search. The theoretical results mentioned in Section V-C do not account for this detail formally, without considering the discretization, and thus we do not provide any results on that effect. However, the parameter associated with this implementational detail is not a critical one and

the NAG search algorithm is robust to ad hoc choice of the parameter for a wide range of environments. In particular, by choosing the bound on the neighborhood search depth to be sufficiently smaller than neighborhood radius, we observe that Proposition 2 can be used to choose the neighborhood radius in nonuniform cost environments.

E. Pathological Cases - Low Cost/Curvature Domains

As explained in Section V-A, comparison of neighborhoods allow distinction between vertices with the same coordinates that are reached via different branches of the search. However, the branching of the search is only possible if the configuration space geometry generates closed geodesics and in particular closed geodesics with enough length so that they can be captured with a reasonable discretization via a proper choice of the neighborhood radius. As the curvature of an environment decreases (e.g., for a high-cost region in a planar domain if the cost is not sufficiently high), the path neighborhoods keep overlapping and is unable to give rise to distinct topo-geometric branches. Instead of creating distinct branches, only a small cusp in the search wave-front is generated, and the search in the neighborhood-augmented graph is unable to compute multiple topo-geometrically distinct paths ($CM=1$ case in Figure 17). Similarly, around a corner of a prism in 3D, the search is only slowed down by a small amount through the corner. This behavior is illustrated in Figure 9(b,d). In these cases, only a single path is found as a result of the search on the neighborhood augmented graph. For such pathological cases, we present an additional algorithm capable of computing geodesic paths around such low cost/curvature domains that do not create multiple branches of the wavefront in the main NAG search algorithm.

1) Cut Point Identification: On a flat planar domain without any holes there is a unique geodesic between any two points and it is neither necessary nor expected to generate different branches of the search. As some small curvature is introduced to the surface by a high-cost region, closed geodesics and correspondingly multiple geodesic paths can arise between two points. However, along the cusp generated in the search wave by the low curvature, these paths are very similar to each other and the corresponding path neighborhood sets are likely to intersect. That is why it is not possible to generate branches of the search in low curvature environments (Figure 9(b)).

Although it is possible to modify the shape and size of the path neighborhood sets, it is not possible to use the theoretical results from Section V-C to come up with a neighborhood radius and heuristic weight that allow distinction between such close paths without causing spurious topo-geometric branches to be generated at other locations in the environment. This is mainly caused by the fact that closed geodesics are either significantly short or non-existent on these pathological environments. A similar behavior is observed around the corners of a 3D prism. Another approach could be to consider a finer discretization, to increase the capability of distinguishing between similar paths. However, we propose an approach to tackle with the low curvature environments, without requiring any changes to discretization or further fine tuning of the neighborhood parameters.

The idea behind the proposed solution is to introduce an artificial *cut* to the neighborhood-augmented graph that will allow separate branches of the search to be generated. To decide where the cut should be introduced, we identify the points along the cusp generated by the low curvature to which the distinct geodesics are separated significant enough. These points will be referred to as *cut points*. Then, it would be possible to treat a region around those points (referred to as *cut point regions* (CPR)) as an obstacle to give rise to different branches of the search.

A pseudocode describing the *cut point* identification and cut point region generation procedures are given in Algorithm 4. When running the search algorithm on low curvature environments, the *cutPointCheck* subroutine is called within the *searchNAG* routine provided in Algorithm 1, after a successor v' is identified the same with an existing vertex w – after Line 20. To identify the cut points algorithmically, we utilize the following observation. Although the path neighborhood sets for vertices at the cusp are not entirely disjoint, they should have relatively smaller intersections (Figure 9(b)). It is possible to compute a neighborhood intersection ratio (ratio of the number of common vertices in the path neighborhood sets to the size of the path neighborhood sets), such that the ratios at a cut point falls below a certain threshold – Line 1 of Algorithm 4. Parallel to that, we also make use of the observation that the lengths of the geodesics to two vertices should be similar to each other at the cusp, which can be confirmed by looking at the difference in g-scores of the vertices – Line 4 of Algorithm 4. We also perform an addition check on the separation between path segments leading to the two vertices (Line 9), following which the CPR is computed using a Dijkstra's search for a fixed radius r_{CP}. The CPR is treated as an obstacle, leading to separate branches to be generated downstream as illustrated in Figure 9(c).

Algorithm 4 Cut Point Check and Cut Point Region Generation Procedure

[bool, CPR]= *cutPointCheck* (w,v',G_N)

Inputs:

a. Two vertices with intersecting neighborhoods, $w \in V_N, v'$ (v' is not inserted in the graph)
b. Parent vertex of v', $v \in V_N$
c. Neighborhood augmented graph, G_N

Output:

a. Boolean (**true** if cut point, **false** if not)
b. Cut point region (a set of vertices), CPR

1: $\alpha =$ *getNeighborhoodIntersectionRatio* $(w.\mathcal{U}, v'.\mathcal{U})$
2: **if** $\alpha > \varepsilon_i$ **then**
3: return (**false**,$\emptyset$)
4: **if** $|g(w.\text{came\_from}) - g(v)| > \varepsilon_g$ **then**
5: return (**false**,$\emptyset$)
6: $p_w =$ *getPathPoint* (w,r_l,G_N) // *getPathPoint* (w,r_l,G_N) returns the vertex at a distance of $r_l \cdot g(w)$ from w, along the shortest path leading up to the start vertex
7: $p_{v'} =$ *getPathPoint* (v',r_l,G_N)
8: $d_{sep} =$ *searchOnGraph* $(p_w,p_{v'},G_N)$ // by running A* from p_w to $p_{v'}$ with a radius of ε_{upper}, returns the distance if $p_{v'}$ is reached, returns **false** otherwise
9: **if** d_{sep} **AND** $d_{sep} > \varepsilon_{lower}$ **then**
10: CPR = *generateCutPointRegion* (w,r_{CP},G_N) // by running Dijkstra's starting from w with a radius r_{CP}, expanded vertices are included within cut point region
11: return (**true**,CPR)

The parameters associated with the cut point identification, neighborhood intersection ratio threshold (ε_i), geodesic length difference threshold (ε_g), geodesic reconstruction portion (r_l), path separation lower and upper bounds (ε_{lower},ε_{upper}), and the cut point region radius (r_{CP}) are to be treated separately from the parameters of the

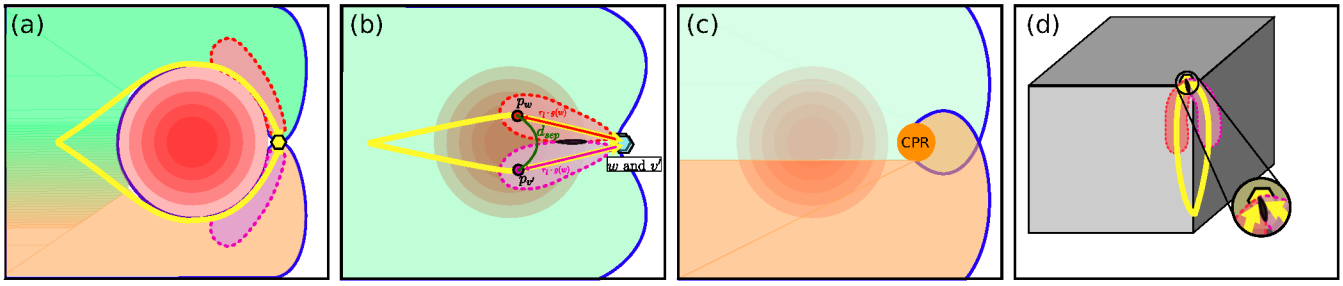


Fig. 9: (a) Through a steep hill (high-cost center with high value) search propagation is significantly slowed down, thus two branches emerge. Neighborhoods remain separate for vertices reached via different branches. (b) Around a flatter hill, (high-cost center with smaller value) neighborhoods maintain an overlap, hence branches do not emerge naturally even on a neighborhood augmented graph (neighborhoods maintain the overlap, because the flatter hill only slows down the search through its center by a small amount, causing a small cusp in the search wave). There exist two vertices w and v' with small neighborhood intersection. **Cut Point Identification:** Shortest paths are constructed until a distance of $r_l \cdot g(w)$. Resulting vertices p_w and $p_{v'}$ have a separation of d_{sep} in between. (c) On a low curvature environment, cut point region is detected at the downstream of the hill. Treatment of these cut points causes branches to emerge around this region. (d) Same/similar behavior is observed around a corner of a 3D prism.

main NAG search algorithm and need to be estimated empirically based on a given low cost/curvature scenario in an environment.

Here we highlight that depending on how the initial and goal configurations are assigned, some of the paths found by the cut point identification based approach might be non-geodesic in the underlying configuration space as they are going around an artificial obstacle introduced by the algorithm. However, it is possible to identify such paths in an automated manner by checking the change in the tangents along the paths near the CPR. A sudden change in the tangent will indicate a non-geodesic path. This behavior is demonstrated by experimental results in Figure 18.

VI. Application to Tethered Robot Path Planning

In this section, we define the tethered robot shortest path planning problem and explain how the neighborhood-augmented graphs can be adapted to solve this problem.

A. Problem Description and Analysis

Tethered robots are mostly deployed to perform tasks in environments where wireless communication is limited and/or robot is unable to operate without an external power source. This work is mainly interested in applications where a tethered spatial robot is to navigate in an environment populated with obstacles. Examples include a drone tethered to an outside base navigating in and around a building with windows or gates or a disaster site with multiple entries and exits, an underwater robot tethered to the surface station navigating in and around a ship wreck or an underwater cave with tunnels and passages. We consider the problem defined as follows:

Definition 7 (Tethered robot shortest path planning problem). *Given a mobile robot (ground, air, underwater) connected to a fixed base via a tether of maximum length L, a start configuration (robot pose and tether configuration) and a goal robot pose, the tethered robot optimal path planning problem seeks to find the globally shortest path that achieves the goal robot pose with some tether configuration without any violations of the maximum tether length constraint.*

For the purposes of this paper, it can be assumed that the tether is made from an elastic material with a maximum length L, which remains taut during the entire operation irrespective of the location of the robot. Alternatively, it can be assumed that there is a tether retraction mechanism at the base, that can remove the remaining slack from the tether when it is not used at its maximum length L. Using any of the two assumptions, a tether configuration can be represented as one of the geodesic paths from the tether base to robot's location. An illustration of a tethered robot in a 2D environment with obstacles is given in Figure 10.

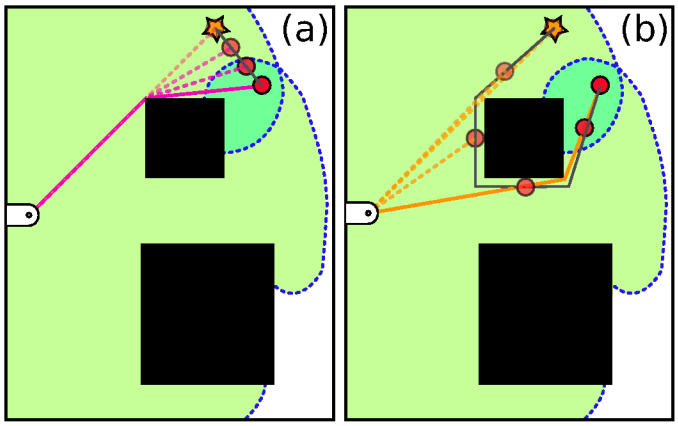


Fig. 10: (a) Tethered robot starts from an initial configuration and can reach the goal via the *globally shortest* path without violating the length constraint. (b) Tethered robot starts from the same pose but with a different tether configuration. From this configuration, the globally shortest path violates the length constraint (it is *untraversible*), thus it reaches the goal via another path that is *geodesic*.

The tether length constraint not only reduces the size of the reachable workspace, but a path planning algorithm for such a system also needs to account for the configuration of the cable. To be able to compute *traversable* paths (paths that do not violate the tether length constraint) for a tethered robot with cable length constraints, it is necessary to build the configuration space graph for the tethered robot, the vertices in which capture not only the pose of the robot, but also the topo-geometric information of the cable configuration. In particular, the same configuration of the robot may correspond to multiple different vertices in the length-constrained tethered configuration space graph based on the configuration of the tether (distinct vertices for topo-geometrically distinct tether configurations, even for the same robot pose). This is illustrated in Figure 10.

B. Method

For solving the tethered robot shortest path planning problem, we adopt an approach similar to [15], where the vertices of a h-augmented graph would maintain information about not only the pose of a robot, but also the homotopy class (h-signature) of the cable. However, this approach cannot be naturally extended to 3D domains since in 3D the homotopy classes of paths do not capture all possible tether configurations. In other words, there may exist geodesic curves that belong to the same homotopy class which cannot be deformed into one another without violating the tether length constraint. As neighborhood-augmented graphs are capable of distinguishing between topo-geometrically distinct paths, they provide a solution to the tethered robot path planning problem applicable to more complex spatial domains. In this work, we use neighborhood-augmented graph to represent the configuration space, the vertices in which not only capture information about the pose of the robot, but also the cable configuration represented by the path neighborhood of the corresponding geodesic path.

Path planning for a tethered robot consists of two stages of search using neighborhood-augmented graphs. The first stage is referred to as *tethered configuration space construction*, where a neighborhood augmented graph is incrementally constructed starting from a base vertex v_b and includes every neighborhood-augmented vertices that are reachable within the tether length constraint. The algorithm used for the construction is identical to the one in Algorithm 1, where the stopping criteria is a distance based one – instead of stopping at a vertex that has matching coordinates with an assigned goal, construction terminates at the first vertex that has a g-score exceeding the maximum tether length. During the construction of this *length-constrained neighborhood augmented graph*, we use the S* algorithm, such that the computed g-scores represent the lengths of the shortest paths in the underlying configuration space.

The second stage is referred to as *length constrained search (LCS)*, where another search is performed on the previously constructed length-constrained neighborhood augmented graph. This search starts from an initial vertex v_s corresponding to the initial robot pose and a unique tether configuration (path neighborhood corresponding to that configuration) in the neighborhood augmented graph. The search terminates when a vertex in the neighborhood-augmented graph with target robot pose/coordinate q_g within the reachable workspace is expanded for any path neighborhood (which would correspond to the final tether configuration). Path resulting from a *LCS* is referred to as *length constrained path (LCP)* and is the shortest path between the start and the goal poses/coordinates that satisfies the tether length constraint starting at the given cable pose.

VII. Experiments and Results

In this section we provide experimental results showing the capabilities of the main NAG search algorithm proposed in Section V(A-D) and its extension using the notion of cut points explained in Section V-E[3]. We also demonstrate results on the application to tethered robot motion planning problem from Section VI. Throughout the experiments, the algorithm is run on an 11th Gen Intel Core i7-1165G7 @ 2.80GHz with 16GB of RAM.

A. Experiments using the Main NAG Search Algorithm

1) Results in 2D and 3D domains: Proposed neighborhood augmented path planner is validated in 6 different planning environments shown in Figure 11. For each environment, the discretization size is specified within the caption as (length $\times$ width) and (length $\times$ width $\times$ height) for 2D and 3D environments respectively. The primary search is performed using S* and the neighborhood searches are performed using A* algorithms. For using the S* algorithm, we discretize the 2D environments into isosceles right triangles with unit side length (2 triangles per unit square) and the 3D environments into uniform tetrahedrons (with 6 tetrahedrons per unit cube). We note that the necessary simplices are generated and the neighborhood augmented graph is constructed on-the-fly as the wavefront propagates. As discussed in Section V, the wavefronts of the search may cross-over during the construction of a neighborhood augmented graph, in which case more than a single simplex can be generated at the same coordinates of the environment corresponding to more than one branch of the wavefront passing over that coordinate. On the other hand, if the wavefront does not reach some parts of the environment, simplices will not be generated for those parts. Therefore, the memory requirement for a neighborhood-augmented search on an environment is directly proportional to the number of expanded vertices, which are reported in the caption of Figure 11 for each environment. Vertices in a 2D environment takes up around 30 kB of storage on average, whereas in 3D they take up around 168 kBs. For each expanded vertex, 2 triangles are generated on 2D environments and 6 tetrahedrons are generated on 3D environments.

On nonuniform cost environments, the cost of an edge lying entirely in white space is equal to the Euclidean norm, whereas an edge lying entirely in the high cost center (marked with darkest shade of red) has a cost scaled by a factor ~ 3. The artifacts in 2D environments (Figure 11(a-c)) are significantly large compared to the discretization, thus the theoretical results provide a large upper bound for the choice of r_n. For computational efficiency, we choose a reasonably small $r_n = 8$ and a reasonably large $\omega = 0.6$. For the 3D environments (Figure 11(d-f)), the upper bounds on r_n are tighter. The shortest closed geodesics in (d-f) have lengths of $20, 20$, and 16 respectively. We choose corresponding r_n's to be 6,6, and 4, accounting for a rollback of 3.

As shown in Figure 11(a-f), neighborhood augmented path planning algorithm can identify topo-geometrically distinct geodesic paths in 2D environments with obstacles, high-cost regions, or a mixture of both. In 3D environments, it can identify distinct paths in the presence of objects with non-zero genus, or even more complex structures such as knots and chains. The local suboptimality of some of the paths obtained in 3D environments are mainly an artifact of the coarse discretization used with S* search algorithm.

Corresponding to each environment in Figure 11, we provide the computation times in the caption. On average, around 68% of the computation time is spent for the neighborhood search and comparison.

2) Comparison between S and A* as the primary search algorithm:* The neighborhood-augmented graph can be searched with any search algorithm, including Dijkstra's, A* and S*. Figure 12 shows the comparison of the results obtained in an 2D domain with non-uniform cost using the S* and the A* search algorithms. The A* search algorithm uses an 8-connected grid-world representation of the domain as the discrete graph, G, using which the neighborhood-augmented graph, G_N, is incrementally constructed for the search and computation of 3 topo-geometrically distinct paths.

3) Comparison with RRT:* Current sampling-based path planning algorithms do not provide a systematic way for computing distinct geodesic paths. To demonstrate the associated challenges, we follow the procedure outlined next. On the environment presented in Figure 11(c), we run the RRT* algorithm (using the implementation

[3]An open source implementation of our algorithm can be found at: https://github.com/asahin1/nbh-aug-planning

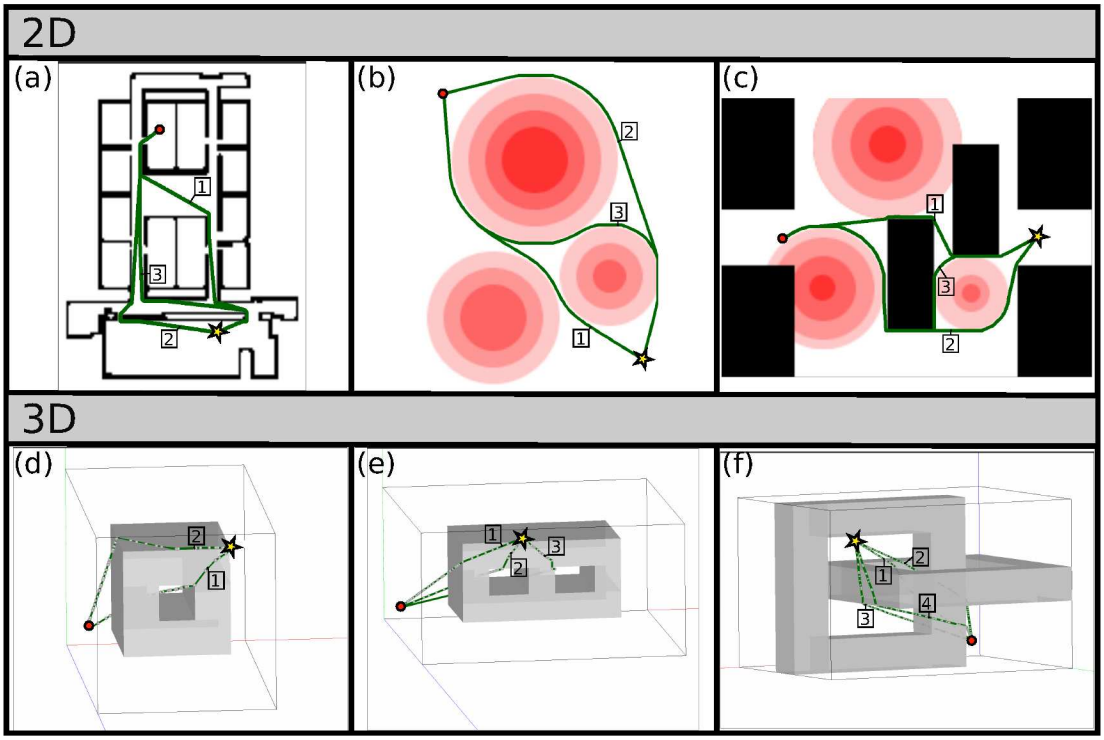


Fig. 11: **Results:** (a) Distinct paths found in a (125×146) 2D building-like environment (expanded 15297 vertices) (3 paths). Path lengths: [141.57,142.72,154.72]. Found in 7 seconds. (b) (150×200) Environment with multiple hills (high-cost centers) (expanded 51779 vertices) (3 paths). Path lengths: [217.93,273.01,280.87]. Found in 35 seconds. (c) (200×150) Environment with a mixture of obstacles and hills (expanded 31191 vertices) (3 paths). Path lengths: [168.48,213.05,231.71]. Found in 20 seconds. (d) $(10\times10\times20)$ 3D environment with genus-1 object (expanded 3585 vertices) (2 paths). Path lengths: [20.74,21.51]. Found in 9 seconds. (e) $(20\times10\times20)$ 3D environment with genus-2 object (expanded 8540 vertices) (3 paths). Path lengths: [20.61,21.30,26.23]. Found in 24 seconds. (f) $(28\times20\times20)$ 3D environment with a chain-like structure (expanded 34247 vertices) (4 paths). Path lengths: [30.97,32.20,32.44,34.53]. Found in 76 seconds.

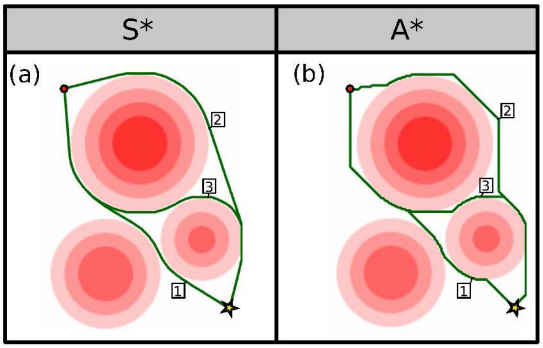


Fig. 12: **Results:** (a) Paths found using S*. Path lengths: [217.93,273.01,280.87]. Found in 35 seconds. (b) Paths found using A*. Path lengths: [223.77,285.83,288.18]. Found in 49 seconds.

available in the Open Motion Planning Library [45]) for 1000 iterations. The number of iterations are limited to increase the likelihood of the algorithm converging into paths other than only the globally optimal one. Using the same settings and starting from an empty tree, the algorithm is run 1000 times, producing 1000 paths. Each run took 0.057 seconds on average. Resulting paths are shown in Figure 13.

As previously shown in Figure 11, the neighborhood-augmented search on the same environment is able to plan 3 distinct geodesics in 20 seconds. In comparison, the total time it takes to generate 1000 paths using RRT* was 57 seconds, in which it failed to find a third path and only was able to plan a path in a distinct homotopy class in one instance. As a result of the reduced number of iterations, paths found by RRT* are also suboptimal with respect to the homotopy classes that they belong to. For a comparison of the path lengths please see the captions of Figures 11(c) and 13.

4) Path planning on spaces with non-Euclidean topology: We also demonstrate the capabilities of the neighborhood augmented path planning algorithm in a space that is not a subset of a Euclidean space. We choose the surface of a cylinder, $\mathbb{S}\times\mathbb{R}$, to demonstrate this capability, with radius r_c and height h. The surface is represented by a $2\pi r_c\times h$ rectangle, with the two vertical edges identified. The search procedure and the resulting paths are shown in Figure 14. We computed 3 paths through search in the neighborhood-augmented graph, although clearly it is capable of finding any number of paths on the cylinder simply by allowing the search to continue further and allowing the search front to wind around the cylinder the required number of times. Results show that the lengths of the paths obtained by the algorithm match with the path lengths computed analytically.

5) Path Planning for a 3-dof planar arm: Neighborhood-augmented path planning can be applied to configuration spaces of other types of robots, including a 3-dof planar arm moving around obstacles, whose free configuration space is a subset of 3-torus ($\mathbb{T}^3$). In this experiment, we use an alternative definition of the cost function, that reflects the energy required to move each joint based upon the lengths of the robot links. The cost of moving joint i by an amount of θ is set equal to the $\theta\sum_{j=i}^{n} l_j^2$, where l_j is the length of link j. Corresponding torus will contain closed geodesics with 3 distinct lengths,

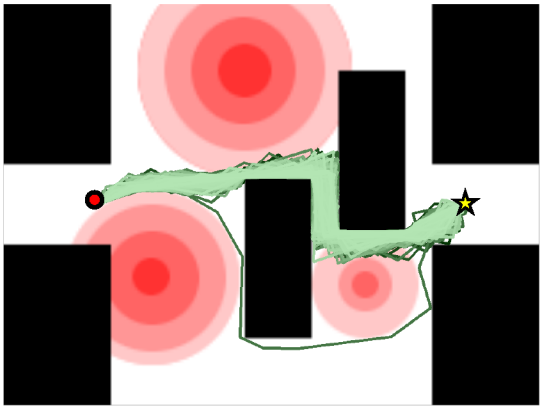


Fig. 13: **Results:** RRT* algorithm is run on an environment with obstacles and high cost regions for 1000 iterations. Within 1000 independent runs of the algorithm, paths from only two distinct homotopy classes are found. 999 out of 1000 times the globally shortest path is found (the top paths), whereas only a single path is found in the other homotopy class that contains the second shortest geodesic path in the environment (the bottom path). The average cost for the top paths are 186.33 and the cost of the bottom path is 221.81.

each corresponding to the motion of one joint. Based on this and the result of Proposition 2, for a planar arm with length links $l_1 = 150, l_2 = 100, l_3 = 50$, we choose $r_n = 35000$, such that the algorithm makes a distinction between paths in which the first two joints go through different motions, but does not differentiate between motions of the last joint. As a result, we obtain 4 distinct paths presented in Figure 15.

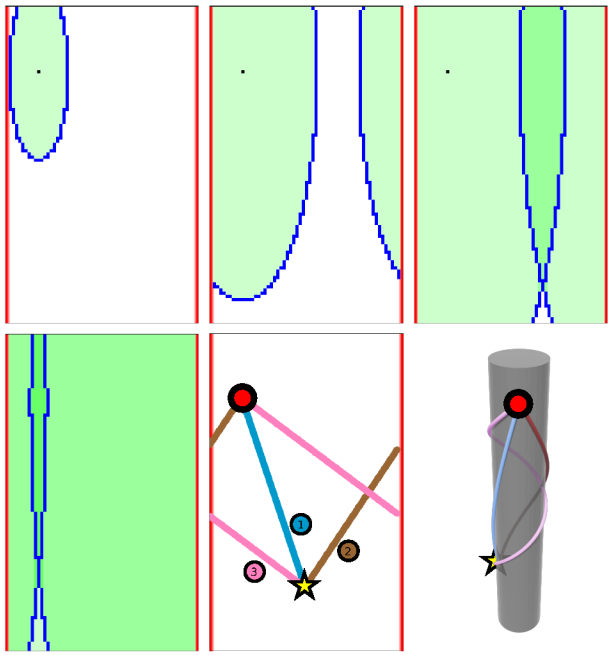


Fig. 14: **Results:** Path planning on a cylindrical surface, $\mathbb{S} \times \mathbb{R}$, with $r_c = 30$ and $h = 100$. The cylinder is represented as a rectangle, whose left and right edges are identified/connected (colored red). Neighborhood-augmented path planning algorithm is used to compute 3 topo-geometrically distinct paths. Path lengths from search: [86.87, 139.25, 258.39]. These path lengths match the analytically computed path lengths.

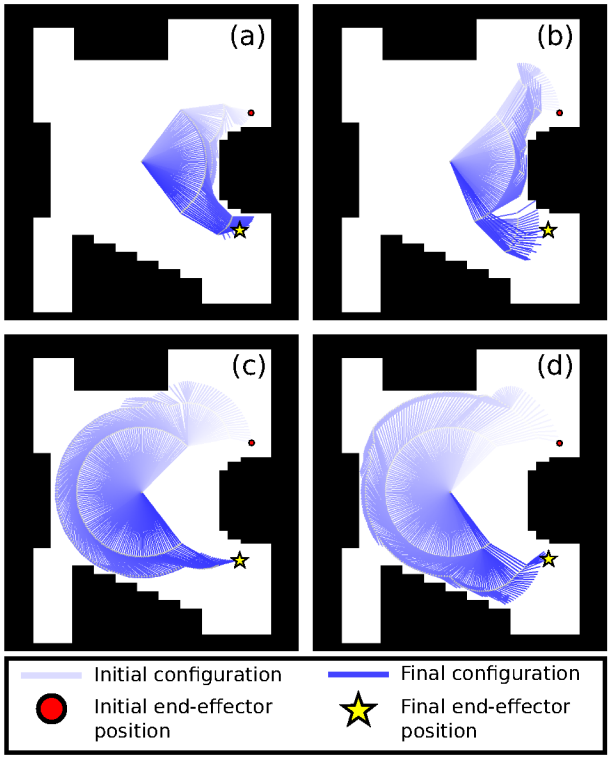


Fig. 15: **Results:** Four distinct sequences of robot motions (paths in the configuration space) for a 3-dof planar arm, between the same initial and goal configurations, corresponding to 4 geodesic paths found via a search using the neighborhood-augmented graph. (a) The first joint follows a clockwise (cw) path, where the last two segments are in an elbow-up configuration, (b) the first joint follows a cw path, where the last two segments are in an elbow-down configuration, (c) the first joint follows a counter-clockwise (ccw) path, where the last two segments are in an elbow-up configuration, (d) the first joint follows a ccw path, where the last two segments are in an elbow-down configuration.

6) Application of Theoretical Results on Configuration Spaces of Various Geometry: We demonstrate the use of Proposition 2 in configuration spaces where the size of the configuration space artifacts and the discretization are of the same order of magnitude so that an ad hoc choice of r_n may not be possible. Results on a 2D environment with an obstacle, a 2D environment with a high-cost region, and on a surface of a cylinder is presented in Figure 16. In each environment, we use the upper bound of $l(c)/2$ to choose r_n, and include some margin of error to account for the implementational inaccuracies.

For the environment with the obstacle, we estimate the length of the shortest closed geodesic using the perimeter of the obstacle. For the high-cost region, we computed the costs of the closed geodesics enclosing discrete cost levels and picked the one with the smallest cost. For the cylindrical environment, we use the circumference of the cylinder to estimate the length of the shortest closed geodesic.

7) Cost multiplier analysis: As mentioned, on uniform cost and flat subsets of Euclidean planes (for example, Figure 11(a)), the edge cost between two vertices is computed using the Euclidean distance as $c(v_i, v_j) = d(v_i, v_j)$, where $d(v_i, v_j) = ||v_i - v_j||_2$. For environments with high-cost regions (as in Figure11(b,c)), the path cost between two vertices is implemented as a function of the color of the corresponding cells and a cost multiplier (CM). Let $\rho(v) : V \to [0,1]$ map the given vertex to the color value. Then the

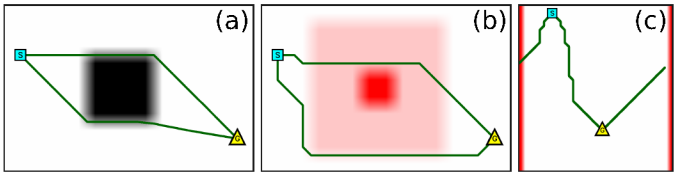


Fig. 16: **Results:** Application of Proposition 2: (a) Two paths are found around a square obstacle where the length of the shortest closed geodesic (equal to the perimeter) is 16 and $r_n = 6$. (b) Two paths are found around a high cost region, where the shortest closed geodesic is around the bright red region with a length of 16 and $r_n = 6$. (c) Two paths are found on the surface of a cylinder with radius 3 (shortest closed geodesic has a length of 6π), where $r_n = 7$.

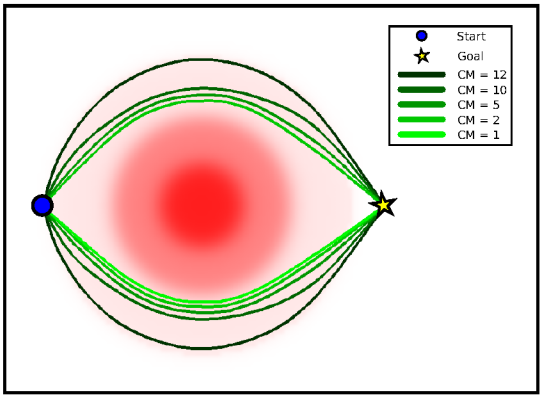


Fig. 17: **Results:** Shortest paths obtained on a nonuniform cost environment. As cost multipliers (CM) decrease, shortest paths tend to get closer to the high cost region. For CM$=1$, the algorithm can only find a single path.

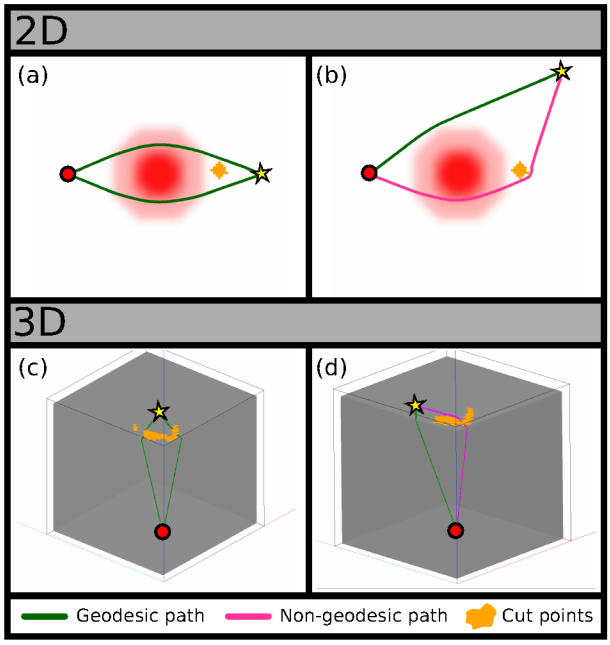


Fig. 18: **Results:** (a) Two geodesic paths around a small *hill* (high cost region, with low cost multiplier), when the start and goal are placed symmetrically. (b) A geodesic and a non-geodesic path around a small hill, with start and goal placed asymmetrically. The non-geodesic path has a sudden change in direction near the cut point region. (c) Two geodesic paths around the corner of a prism, when the start and goal are placed symmetrically. (d) A geodesic an a non-geodesic path around the corner, with start and goal placed asymmetrically. The non-geodesic path goes around the artificial cut introduced by the cut points.

path cost between two vertices in a nonuniform path cost environment is computed as $c(v_i,v_j) = d(v_i,v_j)\left[1+\text{CM}\frac{\rho(v_i)+\rho(v_j)}{2}\right]$.

The geometry of the optimal paths obtained by the neighborhood augmented planning method varies with the cost multiplier being used in the environment. Figure 17 shows how geodesic paths tend to get closer to the center of the high cost region as the cost multiplier decreases. It is worth noting that above a certain value of the cost multiplier paths will not move further away from the high cost region. At the other extreme (if the cost multiplier is below a certain value), the main NAG search will become incapable of distinguishing between multiple optimal paths and return only one of them (see the path for CM$=1$ in Figure 17). This issue was explained further in detail and tackled in Section V-E.

B. Results with Cut Point Identification Based Algorithm

Path-planning using the neighborhood augmented graph with the addition of cut point regions can identify distinct paths in 2D environments with low curvature. In a 3D environment, it can identify distinct path around a corner of a prismatic obstacle. Examples for 2D and 3D scenarios are provided in Figure 18(a-d), for which we arrived at the following set of parameters via tuning: For both 2D and 3D cases, heuristic weight $\omega = 0.6$, $r_l = 0.2$ and $\varepsilon_g = 0.1$ are used. For the 2D case, we use the neighborhood radius $r_n = 8$, $\varepsilon_{lower} = 8$, $\varepsilon_{upper} = 25$, $\varepsilon_i = 0.6$, and $r_{CP} = 3$. For the 3D case, we use the neighborhood radius $r_n = 6$, $\varepsilon_{lower} = 5$, $\varepsilon_{upper} = 10$, $\varepsilon_i = 0.4$, and $r_{CP} = 5$.

In both 2D and 3D cases, a high cost center or a prismatic corner gives rise to two shortest paths, when the start and goal are placed symmetrically with respect to the artifact. However, when placed asymmetrically, the second path can only be identified with the help of the cut points (the artificial cut). Those paths that present abrupt cornering behavior around cut point regions are not considered among the solutions to the k geodesic path planning problem as they are not geodesics in the underlying configuration space.

Based on the experiments shown in Figure 18, around 65% of the computation time is spent on the neighborhood search and comparison and less than 1% of the computation time is spent on the cut point check and cut point region generation on average.

C. Tethered Robot Motion Planning

1) Tether Length Analyis: To demonstrate the effect of the tether length constraint on the paths obtained by the length constrained search algorithm, we consider a building-like environment shown in Figure 19. The building has a total of 3 rooms shaded with red, yellow and green. The large green room spans two floors, whereas the smaller red and yellow rooms are connected via a hole on the floor in between. Both red and yellow rooms are connected to the green room via doors. The green and yellow rooms are accessible from outside via windows. Robot is tethered to a base fixed outside and initially positioned inside the yellow room, which it accessed through the corresponding window. The goal is placed within the green room near the window.

Keeping the initial and goal configurations the same, the length constrained search algorithm is run with varying maximum tether length l. As observed in Figure 19, *LCS* algorithm outputs the shortest path between the start and goal that would satisfy the tether length

constraint. For relatively optimistic/relaxed constraints, resulting path might correspond to the globally shortest path in a given environment. With stricter constraints, the algorithm is still able to find a path between the start and goal. However, these paths might be significantly longer, requiring the robot to exit the building through the windows that it has already used while entering.

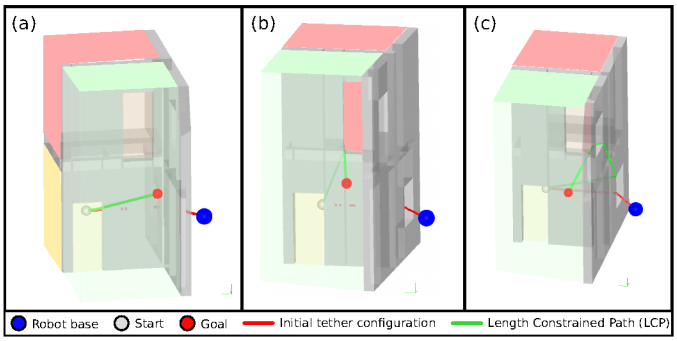


Fig. 19: **Results:** (a) Max. tether length $=30$. Globally shortest path in the environment satisfies the tether length constraint. (b) Max. tether length $=25$. With a shorter tether, robot has to follow the next locally (i.e. under small perturbations of the path) shortest path (which is the second globally shortest path in the environment). (c) Max. tether length $=20$. With a stricter constraint on the tether length, the path that the robot has to follow is significantly longer.

2) Simulated Experiments: We demonstrate the capabilities of the proposed *LCS* algorithm on virtual environments shown in Figure 20 and the accompanying multimedia attachment. In the scenario shown in Figure 20(a), there exists a long rectangular prism in the environment around which there exists geodesic paths within the same homotopy class. The robot is to start from an initial position and a corresponding tether configuration such that the globally shortest path from the start to goal does not satisfy the tether length constraint. The *LCP* obtained from the algorithm is shown using the green line and the sequence of tether configurations are shown using the transparent red lines. As expected, the *LCP* makes the robot move towards its base and go around the other side of the prism. It must be noted that in this scenario the path planning takes place around a genus-0 prism, which was described amongst limiting cases in the previous sections. The difference is that beyond a certain ratio of the length, width and depth of a prism, it is possible to rely on the main NAG search algorithm (proposed in Sections V-A,V-B) to make a geometrical distinction across the faces instead of the corners of the prism.

A similar behavior is observed in Figure 20(b), where the robot has to exit the building-like structure and use the alternative entrance instead of taking the globally shortest path to reach the goal. Figure 20(c) features a trefoil knot shaped obstacle around which a robot is to navigate between a series of goal points. In this scenario, the robot follows the globally shortest paths whenever they satisfy the tether length constraint and uses the alternative locally (i.e. under small perturbations of the path) shortest paths when necessary.

3) Real Robot Experiments: A structure similar to the one shown in Figure 20(b) is constructed using styrofoam blocks for real robot demonstrations. A Crazyflie 2.1 quadrotor platform [46] is connected to a metal hoop that acts as a robot base via a red wool thread that represents the tether. To ensure collision-free flight without having to detect the objects in real-time, we run the path planning algorithm on an environment with inflated obstacles. The discrete output from the path planning algorithm, that is a sequence of 3D points, is supplied to an external trajectory optimizer [47], which is based on the work in [48, 49], to generate a feasible trajectory for the quadrotor. Resulting trajectories are tracked using the controllers proposed in [50] along with an OptiTrack motion capture system within the Crazyswarm framework [51].

The experimental setup and the navigation progress is to be seen in Figure 21 and the accompanying multimedia attachment. The quadrotor is initially placed on the ground closer to the window on the left. The first goal is defined inside the room on the left which is accessed via the window as shown in Figure 21 - Step 2 and 3. The second goal is defined inside the room on the right. The globally shortest path to the second goal is not traversable due to the tether length constraint. As shown in Steps 4-6, the quadrotor has to exit the structure and use the window on the right to reach the goal via a locally (i.e. under small perturbations of the path) shortest path without violating the constraint. The tethered configuration space construction process took 52 seconds and the length constrained search took around 3 seconds for the problem instance used in the real robot experiments.

VIII. Conclusions and Future Work

In this paper, we presented a topo-geometric planning approach that can interface with existing wavefront propagation algorithms to find k geodesic paths. Our method is general in the sense that it provides a way to compute multiple geodesic paths in configuration spaces of different geometry and dimension without requiring any a priori geometric constructions or complex path equivalence checks. Unlike existing topological path planning methods, it is also able to find different geodesic paths that may be homotopic but geometrically distinct due to presence of high cost/curvature regins. Path planning capabilities of the algorithm are demonstrated on a variety of environments and scenarios. We further discussed the limitations of the proposed approach in low-curvature environments and present modifications to the algorithm to increase path planning capabilities in such pathological environments. Lastly, we demonstrated the use of the neighborhood-augmented planning algorithm for a tethered robot path planning problem. For a tethered robot navigating in 3D, we find optimal paths that satisfy the tether length constraints, which we demonstrate using simulations and real robot experiments.

A limitation of our work is the strict dependency on uniform wavefront propagation type algorithms (Dijkstra's or S*) upon which the theoretical results are based. This requires the use of graph search algorithms with no heuristic and poses a computational burden for higher dimensional problems. The theoretical results could potentially be extended to consider algorithms with non-zero heuristic for the primary search to speed-up computation, which would require relaxing the assumptions on the shape of the wavefront. Alternatively, adaptation of the method to sampling-based algorithms can provide a more scalable solution. This adaptation involves further study into strategies for sampling configurations and locally connecting them, while maintaining the neighborhood information. Another future direction is the improvement of the cut point identification based method towards a more robust solution that is parameter independent for the pathological cases presented in the paper.

In its current state, the neighborhood-augmented planning approach provides a general and an effective way of finding multiple geodesic paths on 2- and 3-dimensional configuration spaces, which is proven to be useful in many robotics applications.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CCF-2144246.

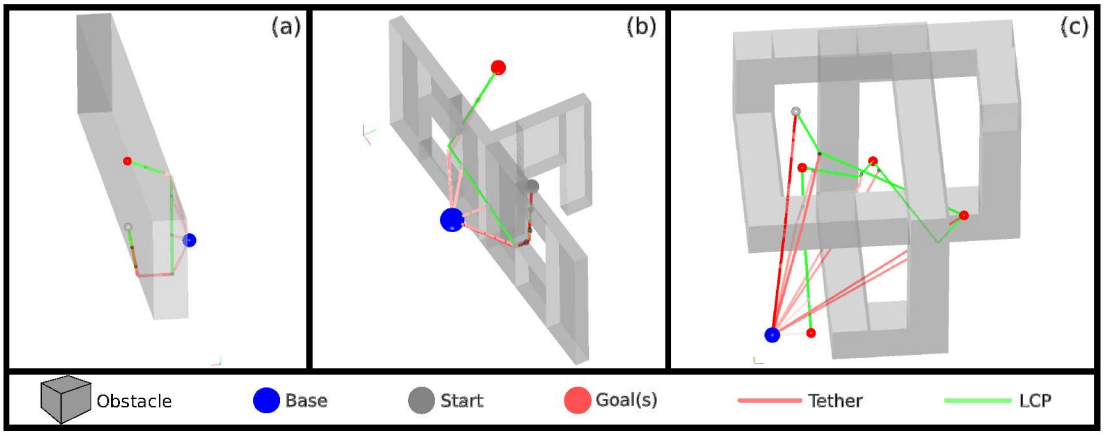


Fig. 20: **Results:** (a) Environment with long rectangular obstacle. (b) Environment with building-like structure. Building has two rooms connected via a door. Both rooms have window access and the robot base is placed outside. (c) Environment with a trefoil knot shaped obstacle. Sequence of tether configurations are plotted starting from opaque to transparent red to highlight the order.

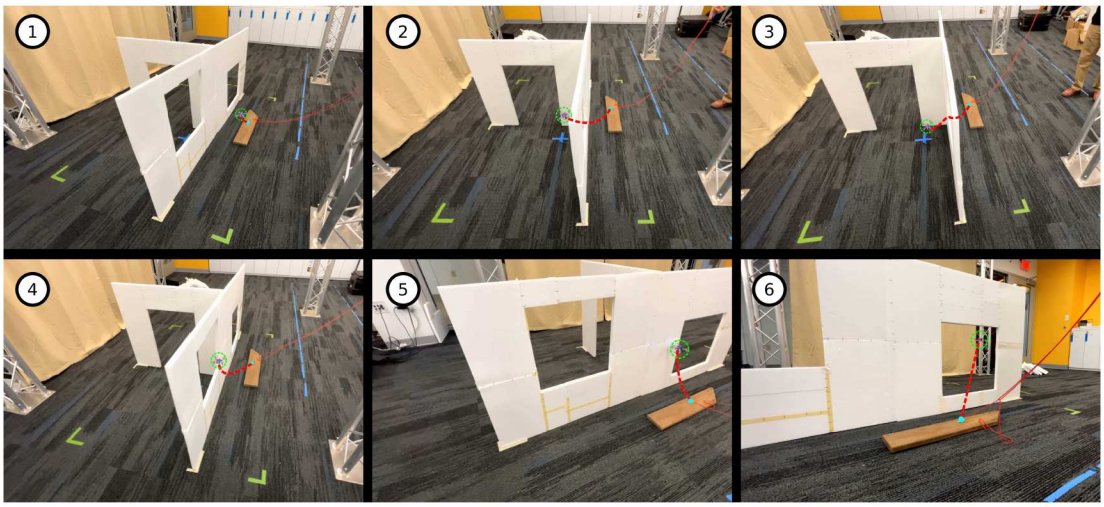


Fig. 21: Photos from real robot experiments for scenario 1a. Steps 1 through 6. Quadrotor, wool tether and robot base are highlighted with colored lines for improved visibility.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/TSSC.1968.300136

[3] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, pp. 1652–1659.

[4] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010. [Online]. Available: https://doi.org/10.1613/jair.2994

[5] M. Cui, D. D. Harabor, and A. Grastien, "Compromise-free pathfinding on a navigation mesh," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 496–502. [Online]. Available: https://doi.org/10.24963/ijcai.2017/70

[6] S. Bhattacharya, "Towards optimal path computation in a simplicial complex," *The International Journal of Robotics Research*, vol. 38, no. 8, pp. 981–1009, 2019. [Online]. Available: https://doi.org/10.1177/0278364919855422

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[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 *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004. [Online]. Available: https://doi.org/10.1109/IROS.2014.6942976

[9] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020. [Online]. Available: https://doi.org/10.1177/0278364919890396

[10] X. Wang, A. Sahin, and S. Bhattacharya, "Coordination-free multi-robot path planning for congestion reduction using topological reasoning," *Journal of Intelligent & Robotic Systems*, vol. 108, no. 3, p. 50, Jul 2023. [Online]. Available: https://doi.org/10.1007/s10846-023-01878-3

[11] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological

exploration of unknown and partially known environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3851–3858. [Online]. Available: https://doi.org/10.1109/IROS.2013.6696907

[12] V. Govindarajan, S. Bhattacharya, and V. Kumar, "Human-robot collaborative topological exploration for search and rescue applications," in *Distributed Autonomous Robotic Systems*. Tokyo: Springer Japan, 2016, pp. 17–32. [Online]. Available: https://doi.org/10.1007/978-4-431-55879-8_2

[13] A. Orthey, B. Frész, and M. Toussaint, "Motion planning explorer: Visualizing local minima using a local-minima tree," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 346–353, 2020. [Online]. Available: https://doi.org/10.1109/LRA.2019.2958524

[14] A. Orthey and M. Toussaint, "Visualizing local minima in multi-robot motion planning using multilevel morse theory," in *Algorithmic Foundations of Robotics XIV*. Springer International Publishing, 2021, pp. 502–517. [Online]. Available: https://doi.org/10.1007/978-3-030-66723-8_30

[15] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1132–1139. [Online]. Available: https://doi.org/10.1109/ICRA.2014.6906996

[16] S. Bhattacharya, S. Kim, H. Heidarsson, G. S. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 799–815, 2015. [Online]. Available: https://doi.org/10.1177/0278364914562236

[17] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, Oct 2012. [Online]. Available: https://doi.org/10.1007/s10514-012-9304-1

[18] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Annals of Mathematics and Artificial Intelligence*, vol. 84, no. 3, pp. 139–160, Dec 2018. [Online]. Available: https://doi.org/10.1007/s10472-018-9596-8

[19] D. Yi, M. A. Goodrich, and K. D. Seppi, "Homotopy-aware rrt*: Toward human-robot topological path-planning," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016, pp. 279–286. [Online]. Available: https://doi.org/10.1109/HRI.2016.7451763

[20] T. Yang, L. Huang, Y. Wang, and R. Xiong, "Tree-based representation of locally shortest paths for 2d k-shortest non-homotopic path planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 553–16 559. [Online]. Available: https://doi.org/10.1109/ICRA57147.2024.10610073

[21] L. Jaillet and T. Simeon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008. [Online]. Available: https://doi.org/10.1177/0278364908098411

[22] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021. [Online]. Available: https://doi.org/10.1109/TRO.2021.3071527

[23] D. Ferguson, C. Baker, M. Likhachev, and J. Dolan, "A reasoning framework for autonomous urban driving," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 775–780. [Online]. Available: https://doi.org/10.1109/IVS.2008.4621247

[24] D. Hong, S. Kimmel, R. Boehling, N. Camoriano, W. Cardwell, G. Jannaman, A. Purcell, D. Ross, and E. Russel, "Development of a semi-autonomous vehicle operable by the visually-impaired," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 539–544. [Online]. Available: https://doi.org/10.1109/MFI.2008.4648051

[25] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2902–2908. [Online]. Available: https://doi.org/10.1109/ROBOT.2010.5509685

[26] S. Swaminathan, M. Phillips, and M. Likhachev, "Planning for multi-agent teams with leader switching," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5403–5410. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139954

[27] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. [Online]. Available: https://doi.org/10.1109/70.508439

[28] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001. [Online]. Available: https://doi.org/10.1177/02783640122067453

[29] C. I. Mavrogiannis and R. A. Knepper, *Decentralized Multi-Agent Navigation Planning with Braids*. Cham: Springer International Publishing, 2020, pp. 880–895. [Online]. Available: https://doi.org/10.1007/978-3-030-43089-4_56

[30] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016. [Online]. Available: https://doi.org/10.1177/0278364915586713

[31] A. Upadhyay, B. Goldfarb, and C. Ekenna, "A topological approach to finding coarsely diverse paths," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6552–6557. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636714

[32] C. Voss, M. Moll, and L. E. Kavraki, "A heuristic approach to finding diverse short paths," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4173–4179. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139774

[33] J.-P. L. T. Siméon and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000. [Online]. Available: https://doi.org/10.1163/156855300741960

[34] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 48–68, 2014. [Online]. Available: https://doi.org/10.1177/0278364912473169

[35] A. Sintov, S. Macenski, A. Borum, and T. Bretl, "Motion planning for dual-arm manipulation of elastic rods," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6065–6072, 2020.

[36] S. Kim and M. Likhachev, "Path planning for a tethered robot using multi-heuristic a* with topology-based heuristics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4656–4663. [Online]. Available: https://doi.org/10.1109/IROS.2015.7354040

[37] S. McCammon and G. A. Hollinger, "Planning and executing optimal non-entangling paths for tethered underwater vehicles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3040–3046. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989349

[38] M. Cao, K. Cao, S. Yuan, T.-M. Nguyen, and L. Xie, "Neptune: Nonentangling trajectory planning for multiple tethered unmanned vehicles," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2786–2804, 2023. [Online]. Available: https://doi.org/10.1109/TRO.2023.3264950

[39] M. Paton, M. P. Strub, T. Brown, R. J. Greene, J. Lizewski, V. Patel, J. D. Gammell, and I. A. D. Nesnas, "Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7034–7041. [On-

line]. Available: https://doi.org/10.1109/IROS45743.2020.9341409
[40] M. Gromov, M. Katz, P. Pansu, and S. Semmes, *Metric structures for Riemannian and non-Riemannian spaces*. Springer, 1999, vol. 152.
[41] M. P. Do Carmo and J. Flaherty Francis, *Riemannian geometry*. Springer, 1992, vol. 6.
[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.
[43] X. Wang and S. Bhattacharya, "A topological approach to workspace and motion planning for a cable-controlled robot in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2600–2607, 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2817684
[44] S. Bhattacharya, R. Ghrist, and V. Kumar, "Multi-robot coverage and exploration on riemannian manifolds with boundaries," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 113–137, 2014. [Online]. Available: https://doi.org/10.1177/0278364913507324
[45] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.
[46] "Crazyflie 2.1 | Bitcraze — bitcraze.io," https://www.bitcraze.io/products/crazyflie-2-1/.
[47] "GitHub - whoenig/uav_trajectories: Helper scripts and programs for trajectories," https://github.com/whoenig/uav_trajectories.
[48] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Cham: Springer International Publishing, 2016, pp. 649–666. [Online]. Available: https://doi.org/10.1007/978-3-319-28872-7_37
[49] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1872–1878. [Online]. Available: https://doi.org/10.1109/IROS.2015.7353622
[50] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525. [Online]. Available: https://doi.org/10.1109/ICRA.2011.5980409
[51] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989376

APPENDIX

Proof of Proposition 1: We prove the equivalent contrapositive statement: *If* $\mathcal{U}_1 \cap \mathcal{U}_2 \neq \emptyset$*, then sep*$(\gamma_1', \gamma_2') \leq \frac{4r_n}{1-\omega}$.

Suppose $u \in \mathcal{U}_1 \cap \mathcal{U}_2$ as shown in Figure 8. Since the path neighborhood sets are constructed using A* search with radius r_n and with the g-score of the primary search as the heuristic function with some heuristic weight ω, for any $u \in \mathcal{U}_i$ we have

$$d(u,x_i) + \omega d(u,v_s) \leq r_n + \omega(r_s - r_n) \quad i = 1,2 \tag{1}$$

and

$$d(u,x_i) \leq r_n \quad i = 1,2 \tag{2}$$

Since $\gamma_i' = \gamma_i \cap \mathcal{U}_i$ (given by the green and red colored lines in Figure 8) and γ_i (given by the black colored lines extending up to v_s) is the shortest path connecting x_i in the open list and v_s, for any $v \in \gamma_i'$ we have

$$d(v,v_s) + d(v,x_i) = r_s \tag{3}$$

Substituting Equation 3 into Inequality 1,

$$\begin{aligned} d(u,x_i) + \omega d(u,v_s) &\leq r_n + \omega(d(v,v_s) + d(v,x_i) - r_n) \\ d(u,x_i) + d(v,x_i) &\leq r_n(1-\omega) + (1+\omega)d(v,x_i) \\ &\quad + \omega(d(v,v_s) - d(u,v_s)) \end{aligned} \tag{4}$$

From the definition of γ_i',

$$d(v,x_i) \leq r_n \tag{5}$$

Using the following triangle inequality,

$$|d(v,v_s) - d(u,v_s)| \leq d(v,u) \leq d(u,x_i) + d(v,x_i) \tag{6}$$

Inequality 4 reduces to

$$\begin{aligned} d(v,u) &\leq r_n(1-\omega) + (1+\omega)r_n + \omega d(v,u) \\ (1-\omega)d(v,u) &\leq 2r_n \end{aligned} \tag{7}$$

For $\omega \in [0,1)$,

$$\begin{aligned} sep(\gamma_1', \gamma_2') &\leq d(v_1,v_2) \leq d(v_1,u) + d(v_2,u) \\ &\leq 2d(v,u) \leq \frac{4r_n}{1-\omega} \end{aligned} \tag{8}$$

□

Proof of Proposition 2: We prove and utilize the following Lemmas for the proof.

Lemma 1. *If* $d(x_1,x_2) > 2r_n$*, then* $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$.

Proof. We prove the equivalent contrapositive statement: *If* $\mathcal{U}_1 \cap \mathcal{U}_2 \neq \emptyset$*, then* $d(x_1,x_2) \leq 2r_n$. Suppose $u \in \mathcal{U}_1 \cap \mathcal{U}_2$ as shown in Figure 8. Since the path neighborhood sets are constructed using A* search with radius r_n, for any $u \in \mathcal{U}_i$ we have

$$d(u,x_i) \leq r_n \quad i = 1,2 \tag{9}$$

From triangle inequality,

$$d(x_1,x_2) \leq d(u,x_1) + d(u,x_2) \leq 2r_n \tag{10}$$

□

Lemma 2. *For any pair of vertices* x_1, x_2 *at the same configuration, reached via topo-geometrically distinct paths,* $d(x_1,x_2) \geq l(c^*)$.

Proof. Since x_1, x_2 are at the same configuration, any path between x_1, x_2 in the closed list will be a closed path on the configuration space. By definition, any closed path on the configuration space is longer than the shortest closed geodesic c^*. Then $d(x_1,x_2) \geq l(c^*)$. □

From Lemmas 1 and 2,

$$r_n < \frac{l(c)}{2} \implies d(x_1,x_2) > 2r_n \implies \mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset \quad \square$$

Alp Sahin is a Ph.D. student in the Department of Mechanical Engineering and Mechanics at Lehigh University, PA, U.S.A. He received his B.Sc in Mechanical Engineering from Bogazici University, Istanbul, in 2014. He received his M.S. in Robotics Engineering from Worcester Polytechnic Institute (WPI), MA, U.S.A, in 2021 with a thesis on motion planning for in-hand manipulation. Currently, his research focuses on the application of geometrical and topological methods to graph-based motion planning for robots.

Subhrajit Bhattacharya is an associate professor in the Department of Mechanical Engineering and Mechanics of Lehigh University, PA, U.S.A. He received his Ph.D. in Mechanical Engineering and Applied Mechanics in 2012, and was a postdoctoral researcher in the Department of Mathematics until 2016, at the University of Pennsylvania, PA, U.S.A. His research interests are centered around motion planning and control of autonomous, intelligent and networked robotic systems using topological and geometric methods. He is a recipient of the 2022 NSF CAREER award.