This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



# Narrow Passage Path Planning via Homotopy-Preserving Collision Constraint Interpolation

Minji Lee, Jeongmin Lee and Dongjun Lee†

***Abstract*—Narrow passage path planning is a prevalent problem from industrial to household sites, often facing difficulties in finding feasible paths or requiring excessive computational resources. We propose a Homotopy Optimization Method (HOM) tailored for the narrow passage problem, utilizing a novel collision constraint interpolation method using signed distance functions (SDF). The framework begins by decomposing the environment into convex objects and representing it as a simplicial complex based on their connectivity. This representation enables topological analysis to induce a easy-to-hard sequence of collision constraint interpolation that preserves homotopy equivalence. Using this collision constraint interpolation, the optimization proceeds through a series of subproblems, gradually guiding the path to the final solution. Several examples are presented to demonstrate how the proposed framework addresses narrow passage path planning problems.**

***Index Terms*—Motion and path planning, manipulation planning, nonholonomic motion planning.**

## I. Introduction

Path planning is a fundamental and crucial aspect of robotic tasks. Despite its long-standing significance, path planning in narrow passage remains particularly challenging with active researches continuing even to this day [1], [2], [3]. Such environments are not merely common in industrial settings, but are often encountered in everyday scenarios, such as tight assembly tasks or navigating cluttered spaces.

Historically, most path planning techniques have leaned on sampling-based methods such as rapidly-exploring random tree (RRT) [4] or probabilistic roadmap (PRM) [5]. These methods offer significant advantages, including probabilistical completeness [6], and a straightforward problem formulation, which generally requires only the computation of path length and collision checks. Despite these benefits, they often face challenges in narrow passages, which can lead to significant sampling inefficiencies [2].

More recently, optimization-based path planning has emerged as another promising approach [7], [8], [9], [10], framing the path planning problem as an optimization. These methods leverage gradient information to quickly converge to low-cost paths in terms of path length, or other task specific metrics, and also offer greater flexibility in integrating various cost and constraints. Despite their advantages, the inherent non-convexity of collision avoidance often leads to local minima [9], particularly in narrow passages where deep penetration is more likely. Deep penetration poses challenges because contact features such as witness points and contact normals are only well-defined when there is no contact or minimal penetration [11]. Moreover, in situations of deep penetration, especially when the path passes through the medial axis [12], the contact normals between adjacent waypoints may become inconsistent, thus pushing the optimizations in opposite directions, making it difficult to escape from infeasibility [9].

One effective approach to address the challenges in such complex nonlinear optimization is the homotopy optimiza-

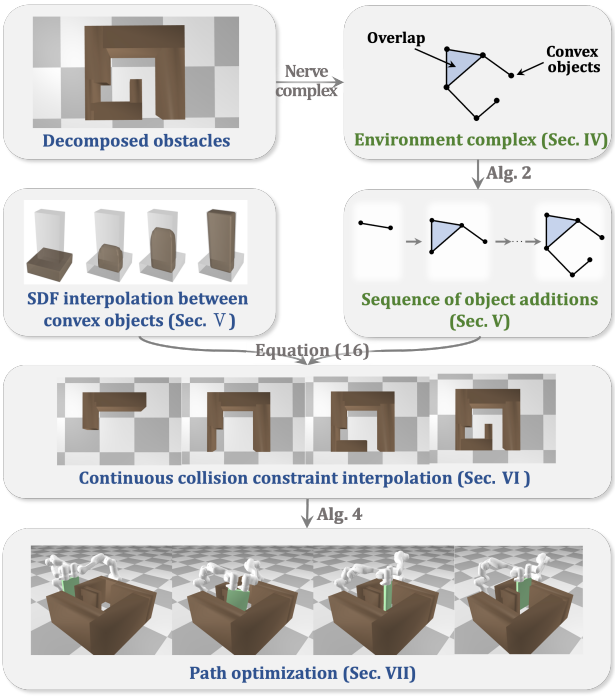



Fig. 1: Overview of the proposed framework. First, from a given convex decomposed obstacles in environment, an environment complex is constructed from their nerve complex. Next, a sequence of object additions is determined based on the environment complex. Finally, path optimization is performed using homotopy optmization, where each subproblem is defined by a continuous interpolation of the collision constraints as the convex objects are gradually introduced into the environment according to the sequence.

This research was supported by Samsung Research, the Ministry of Trade, Industry & Energy (MOTIE) of Korea (RS-2024-00419641), and the National Research Foundation (NRF) funded by the Ministry of Science and ICT (MSIT) of Korea (RS-2022-00144468).

The authors are with the Department of Mechanical Engineering, IAMD and IOER, Seoul National University, Seoul, Republic of Korea, 08826. {mingg8,ljmlgh,djlee}@snu.ac.kr. Corresponding author: Dongjun Lee.

tion method (HOM) [13]. HOM addresses such problems by transforming the original problem into a series of simpler subproblems, progressively modifying them to approximate the original formulation.

However, during this transformation, the solution space may undergo significant topological changes, such as the emergence of abbreviated paths or folds [14]-which may lead the optimizer to become stuck in infeasible regions. Recognizing this, we propose a narrow passage path planning framework that ensures topological consistency throughout the homotopy process by *aligning optimization-level homotopy transformations with geometric-level homotopy equivalence of the collision environment*. Starting with relaxed collision constraints, HOM facilitates the optimization for early-stage subproblems, and the constraints are then gradually restored to their original form, guiding the solution toward the goal. Crucially, our framework guarantees that the transformation preserves the homotopy equivalence of the occupied space in Euclidean space. By maintaining this topological structure, we reduce disruptions in the solution space and significantly improves the likelihood that the path can be continuously updated and remain feasible as the constraints are tightened.

To build this framework, our initial goal is to *develop a method of homotopy-preserving collision constraint interpolation using a signed distance function (SDF)*. This process begins by decomposing the obstacles in the environment into convex objects. The subproblem sequence is constructed by starting with a simplified environment containing only a subset of the convex objects. Subsequent subproblems are created by sequentially introducing additional convex objects into the environment. At each step, the addition of new objects is carried out through interpolation between the SDFs of the newly added objects and the existing objects in such a way that preserves the homotopy of the environment. Since direct interpolation of SDFs can create sharp surface transitions, we apply a shaping function to smooth the resulting geometry. This smoothing reduces abrupt changes in contact normals, thereby improving optimization stability. To achieve this homotopy-preserving constraint interpolation using SDF, we analyze the topology of the environment using a simplicial complex, which we refer to as an *environment complex*, and derive the necessary conditions for the interpolation to preserve homotopy.

Our second objective is to *solve the narrow passage path planning problem via HOM using the collision constraint interpolation*. The optimization progresses through a series of subproblems, gradually restoring the simplified collision constraints back to the original form while ensuring the path satisfies the intermediate constraints at each step. This process involves several key techniques including collision detection of interpolated collision constraints, adaptive adjustment of the interpolation variable, and continuous collision avoidance. By systematically integrating these techniques, we can carry out the optimization using the interpolated collision constraints.

A recent study [15] introduced a narrow passage path planning method based on collision constraint interpolation. However, its applicability is limited to environments composed of convex objects with tree-like connectivity. Additionally, the method suffers from inefficiencies due to challenges in broad-phase collision detection and the rigid, increment-based update of the interpolation variable, which can result in unnecessary computations and difficulties in selecting an appropriate increment. Furthermore, the method does not explicitly handle continuous collision avoidance, which may cause intermediate collisions to go undetected when collision checks are performed only at discrete waypoints. Our work addresses these limitations through the following contributions:

- An analysis on how the shaping function alleviates the sharpness of the interpolated environment is presented in Section V.
- In Section VI, we introduce the concept of an *environment complex* to characterize the environment. This approach allows for a broader range of environments, and we provide a detailed analysis and rigorous proof concerning the preservation of homotopy equivalence.
- In Section VII-A, a new broad-phase collision detection method for the interpolated collision constraints is proposed. Here, we introduce a novel concept called the *interpolated support function*.
- Furthermore, an adaptive method for adjusting the interpolation variable to enhance efficiency is provided in Section VII-B.
- We propose a path refinement method to facilitate continuous collision avoidance in Sec. VII-C.

The rest of the paper is organized as follows. Section II reviews related works, followed by Section III, which introduces the notations used throughout the paper and provides mathematical preliminaries. Section IV presents the problem formulation and outlines the assumptions underlying our framework. In Section V, we propose the SDF interpolation method between convex objects and extend it to homotopy-preserving collision constraint interpolation. Section VII describes the optimization algorithm, which leverages the collision constraint interpolation. Path planning examples using our proposed framework, along with some comparative analyses, are presented in Sec. VIII. Finally, Section IX concludes the paper with some summary and closing remarks.

## II. Related Works

### A. Narrow Passage Path Planning

Sampling-based path planning methods have long been a primary solution to the planning problems. However, these methods often struggle to efficiently explore narrow free spaces in configuration spaces. To address this issue, non-uniform guided sampling strategies have been developed to improve sampling efficiency. Some approaches guide sampling based on task objectives or exploration progress [16], [17]. While effective in many cases, these methods can struggle in extremely narrow passages as they do not explicitly account for the geometry information. Other approaches have exploited obstacle geometry to improve sampling near obstacles or within narrow passages-for example, sampling through medial axis [18], retracting samples to free space [19], guiding samples near surfaces using Gaussian sampling [20], and directing tree growth toward obstacle boundaries [21]. These strategies, by leveraging obstacle-induced structure, aim to address the limitations of uniform random sampling in complex

environments. More recently, learning-based guided sampling techniques have been introduced to enhance the sampling performance. These include learning sampling distribution from demonstrations [22], selecting which pair of trees to connect [23], or learning connectivity online during planning [2]. Despite these advancements, a fundamental limitation of sampling-based methods remains: they typically rely on the availability of a feasible goal configuration, however, identifying such a configuration is inherently challenging in narrow passages.

Another branch of path planning research focuses on optimization-based path planners, which formulates the path planning problems as optimization. Techniques such as CHOMP [7], STOMP [10], and TrajOpt [9] provide high-quality paths than sampling-based methods, but are highly prone to local minima, particularly in narrow passages where deep penetration into obstacles is more likely. In such cases, the difficulty in well-defining contact features and the inconsistencies in contact features between adjacent waypoints can cause the optimization to fail, resulting in a trajectory that becomes stuck in infeasibility.

To address the problem of local minima, recent literature [24] proposed a method that formulates path planning as a mixed-integer programming and applies convex relaxation to efficiently find global solutions. The approach leverages Graph of Convex Sets (GCS) to represent the configuration space and solve the relaxed problem efficiently. Yet, when faced with the narrow passage, constructing such GCS shares some sampling inefficiencies observed in the sampling-based methods. Algorithms such as C-IRIS [25] are used to generate these graphs, which involve iteratively sampling a collision-free configuration and expanding convex region around it. This process requires repeated high-dimensional sampling and collision checking, both of which becomes increasingly inefficient in narrow passage environments [26]. In contrast, our approach performs convex decomposition directly on the obstacles in Euclidean space, which avoids sampling in configuration space and repeated collision checking. As a result, it remains efficient even in narrow passage environment.

### B. Homotopy Optimization Method

Homotopy optimization method is an approach that addresses complex optimization problems by progressively solving a series of subproblems, starting from a simpler version and gradually increasing the complexity, transforming into the original problem [27]. This approach has been adopted to various optimization challenges in robotics due to its effectiveness in managing complex constraints. Especially, various collision-free motion planning problems have been addressed by adjusting model parameters [14], contact dynamics [28], position of obstacles [29], and collision avoidance thresholds [30], [31] to create more manageable subproblems.

As the complexity of the problem increases, however, the solution space can undergo significant changes, leading to more complicated topology of solution space [14]. These changes can disrupt the transition from intermediate solutions to the final desired path, especially when there are large shifts in the solution space between successive subproblems. To circumvent this issue, [14] proposes a probabilistic approach, wherein the homotopy parameter is sampled to prevent the optimization from becoming stuck in local minima within intermediate subproblems. However, this approach has ambiguity in determining which variable quantity should be used as the homotopy parameter and and how it should be initialized to simplify the problem.

Our methodology adopts a different approach to address the problem of changes in the solution space. By relaxing collision constraints to generate subproblems, we ensure that the occupied space defined by these constraints maintains homotopy equivalence throughout the transformation process. This approach mitigates issues arising from changes in the solution space, such as the emergence of abbreviated paths, while also circumventing the challenge of selecting a specific variable to serve as the homotopy parameter.

### C. Shape Interpolation

Shape interpolation methods generally aim to generate plausible transitions between multiple shapes. The technique is widely adopted in various computer graphics applications, including animation and motion pictures. These methods typically involve selecting specific geometric quantities (e.g., mesh or implicit function), interpolating these quantities between the shapes, and then reconstructing the shapes from the interpolated quantities [32].

Traditional mesh-based interpolation selects mesh and vertex points as the geometric quantities, construct transformation matrices between the nodes of the shapes, and extract the morph from these matrices [33]. However, these methods often require consistent tetrahedral meshes of the shapes, which is not always achievable, and may result in undesirable artifacts such as large distortions [34]. Beyond mesh-based quantities, implicit distance function-based quantities have also been explored for shape interpolation. These include interpolation via weighted Minkowski summation [35], Wasserstein distance over space probability [36], and smoothed SDF interpolation using variational implicit function [37].

While traditional shape interpolation methods in computer graphics focus on visually seamless transitions and preserving geometric properties, they are not inherently designed to maintain topological consistency. This limitation hinders their applicability in HOM, as topological inconsistencies can significantly alter the solution space, complicating path planning. Our method overcomes this limitation by introducing homotopy-preserving interpolation using SDF, tailored for efficient path planning.

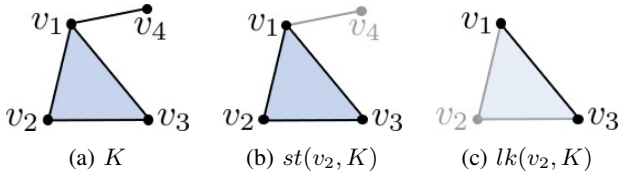
## III. Notations and Mathematical Preliminaries

### A. Signed Distance Function and Occupied Space

The Signed Distance Function (SDF) for an object $v$ in $\mathbb{R}^3$ is a function that quantifies the distance from any point $x \in \mathbb{R}^3$ to the surface of the object. Denoted as $\mathrm{sd}_v(\cdot) : \mathbb{R}^3 \to \mathbb{R}$, it is formally defined as:

$$\mathrm{sd}_v(x) = \begin{cases} -\inf\limits_{y \in \partial v} d(x, y) & \text{if } x \in v \\ \inf\limits_{y \in \partial v} d(x, y) & \text{else} \end{cases}$$

where $\partial v$ is the boundary of $v$, and the metric $d(x, y)$ represents the Euclidean distance between point $x$ and $y$. For




(a) $K$ (b) $st(v_2, K)$ (c) $lk(v_2, K)$

Fig. 2: Example of a simplicial complex consists of four vertices $\{v_1, v_2, v_3, v_4\}$, and a star and a link of a vertex $v_2$.

a collection of objects $\mathcal{V} = \{v_1, \ldots, v_n\}$, the combined SDF is given by $\mathrm{sd}_{\mathcal{V}}(x)$, defined as:

$$\mathrm{sd}_{\mathcal{V}}(x) := \min(\mathrm{sd}_{v_1}, \ldots, \mathrm{sd}_{v_n})$$

Let us define an occupied space of a function $d(\cdot) : \mathbb{R}^3 \to \mathbb{R}$ as a set corresponds to the region in $\mathbb{R}^3$ where function $d$ is non-positive:

$$\mathcal{O}(d) = \left\{x \in \mathbb{R}^3 \mid d(x) \leq 0\right\}$$

Here, $d(x) = 0$ implicitly defines the surface of the occupied space. For a point $x \in \mathbb{R}^3$ lying on the surface $d(x) = 0$, the normal vector at the point, denoted as $n(x)$, is given by the gradient $\nabla d(x)$:

$$n(x) := \frac{\nabla d(x)}{\|\nabla d(x)\|} \tag{1}$$

The occupied space of the SDF of a collection of objects $\mathcal{V}$, denoted as $\mathcal{O}_{\mathcal{V}}$, is exactly the union of the occupied space of the individual objects:

$$\mathcal{O}_{\mathcal{V}} := \mathcal{O}(\mathrm{sd}_{\mathcal{V}}) = \bigcup_{v \in \mathcal{V}} v$$

Furthermore, for any functions $g_1(\cdot), g_2(\cdot) \in \mathbb{R}^3 \to \mathbb{R}$, the following property holds:

$$\mathcal{O}(\min(g_1, g_2)) = \mathcal{O}(g_1) \cup \mathcal{O}(g_2) \tag{2}$$

### B. Abstract Simplicial Complex

A *simplicial complex* is a standard concept in algebraic topology, defined as a mathematical structure composed of simplices [38]. A *simplex* is the convex hull of finite sets of points in a Euclidean space. Depending on the number of points, a 0-simplex is a single point, a 1-simplex is a line segment connecting two points, a 2-simplex is a filled triangle formed by three points, and so on. Each simplex also contains lower-dimensional simplices as its *faces*. For example, a triangle (2-simplex) has faces of vertices (0-simplices) and edges (1-simplices). A simplicial complex is a set of simplices that satisfies the following conditions:

1) *Face condition*: If a simplex is a part of the complex, all its faces must also be part of the complex.
2) *Intersection condition*: If two simplices intersect, their intersection must be a face of both simplices.

Building on this concept, the well-established notion of an *abstract simplicial complex* generalizes this idea by focusing on the combinatorial relationships between vertices, rather than their geometric representation [38]. Formally, an abstract simplicial complex $K = (\mathcal{V}, \Sigma)$ is defined as a pair of a finite set of vertices $\mathcal{V}$ and a simplices $\Sigma$, which is subsets of $\mathcal{V}$, that satisfies:

$$\sigma \in \Sigma,\ \tau \subseteq \sigma,\ \tau \neq \varnothing \ \Rightarrow\ \tau \in \Sigma \tag{3}$$

where each simplex $\sigma \in \Sigma$ is a nonempty finite subset of vertices, and $\tau \in \Sigma$ denotes any nonempty subset of $\sigma$. This condition (3) mirrors the face condition of a simplicial complex. In this paper, the term *complex* specifically refers to an *abstract simplicial complex* for simplicity.

A *closed star* of a vertex $v$ in $K = (\mathcal{V}, \Sigma)$ is a subcomplex defined as follows:

$$st(v, K) := \{\sigma \in \Sigma \mid \sigma \cup \{v\} \in \Sigma\}$$

The *link* of $v$ in $K$ is defined as the set of simplices in $st(v, K)$ that do not contain $v$:

$$lk(v, K) := \{\sigma \in st(v, K) \mid v \notin \sigma\}$$

A *simplicial cone* with vertex $v$ over a complex $K = (\mathcal{V}, \Sigma)$ is defined as:

$$vK := \{\{v\}, \tau \mid \tau \in \Sigma \text{ or } \tau = \sigma \cup \{v\}, \sigma \in \Sigma\}$$

where $v$ is a vertex not included in $\mathcal{V}$. The operation of *deleting a vertex* $v$ from a complex $K$ is defined as:

$$K - v := \{\sigma \in \Sigma \mid v \notin \sigma\}$$

**Example** Consider the following abstract simplicial complex $K$ with vertices $v_1, v_2, v_3, v_4$ and the following simplices:

$$K = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}, \{v_1, v_2, v_3\}\}$$

which can be seen that it satisfies (3), and can be visualized as Fig. 2a. In the figure, each vertex represents a simplex of cardinality one, each edge represents a simplex of cardinality two, and each triangle represents a simplex of cardinality three.

For vertex $v_2$, the closed star $st(v_2, K)$ and the link of $v_2$ can be represented as:

$$st(v_2, K) = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}, \{v_1, v_2, v_3\}\}$$
$$lk(v_2, K) = \{\{v_1\}, \{v_3\}, \{v_1, v_3\}\}$$

which can be visualized as Fig. 2b and Fig. 2c.

### C. Homotopy, Homotopy Equivalence, and Homotopy Optimization Method

**Definition 1** (Homotopy)**.** *A homotopy between two continuous functions $f, g : X \to Y$ is a continuous map $H : X \times [0, 1] \to Y$ such that $H(x, 0) = f(x)$ and $H(x, 1) = g(x)$ for all $x \in X$.*

Intuitively, a homotopy provides a smooth deformation from $f$ to $g$. When we extend the idea of homotopy to topological spaces, we obtain the notion of homotopy equivalence, which captures a fundamental similarity between spaces based on their deformability.

**Definition 2** (Homotopy equivalence)**.** *Two topological spaces $X$ and $Y$ are said to be homotopy equivalent if there exist continuous maps $f : X \to Y$ and $g : Y \to X$, such that the*

*compositions $g \circ f$ and $f \circ g$ are homotopic to the identity maps on $X$ and $Y$, respectively. In this case, $X$ and $Y$ are homotopy equivalent, denoted $X \simeq Y$.*

Intuitively, two spaces $X$ and $Y$ are homotopy equivalent if one can be transformed into the other through operations such as bending, stretching, or shrinking, without any tearing or gluing. Spaces that are homotopy equivalent to a point is called *contractible*. A transformation between spaces is said to be *homotopy-preserving* if it maintains the homotopy equivalence between the initial and final spaces.

The concept of homotopy can also be applied to optimization to solve a challenging nonlinear optimization problem.

**Definition 3** (Homotopy optimization method)**.** *The homotopy optimization method (HOM) addresses optimization problems by continuously transforming a problem into a simpler form and then progressively restoring it to the original form. For an optimization problem $F(x)$ over a feasible set $x \in X$, HOM constructs a homotopy $H(x, \alpha)$, where $\alpha \in [0, 1]$ is a continuation parameter, such that:*

1) *$H(x, 0)$ is a simplified optimization problem that is easy to solve: $H(x, 0) = F_{easy}(x)$.*
2) *$H(x, 1)$ corresponds to the original optimization problem: $H(x, 1) = F(x)$.*
3) *$H(x, \alpha)$ smoothly transitions between $H(x, 0)$ and $H(x, 1)$.*

Despite the shared terminology, it is important to note that the term "homotopy" in HOM is not directly related to the topological concept of homotopy equivalence. Instead, it refers to the idea of a continuous transformation parameterized by $\alpha$.

### D. Nerve Complex

A nerve complex of a set is an abstract simplicial complex that represents the structure of how sets in a collection intersect [38].

**Definition 4** (Nerve complex)**.** *Consider a collection of sets $\mathcal{V} = \{v_i\}_{i \in \mathcal{I}}$ indexed by $\mathcal{I}$. The nerve of $\mathcal{V}$, denoted as $N(\mathcal{V})$, is an abstract simplicial complex where set of all finite subsets $J \subseteq \mathcal{I}$ such that the intersection of the corresponding sets $v_j$ for $j \in J$ is non-empty:*

$$N(\mathcal{V}) := \left\{ J \subseteq \mathcal{I} \mid \bigcap_{j \in J} v_j \neq \varnothing \right\}$$

The nerve complex $N(\mathcal{V})$ can in certain cases accurately represent the topology of $\mathcal{V}$. The following theorem provides sufficient conditions for $\mathcal{V}$ guaranteeing that $N(\mathcal{V})$ represents the topology of $\bigcup_{i \in \mathcal{I}} v_i$.

**Theorem 1** (Nerve theorem [38])**.** *Consider a given family of sets $\mathcal{V} = \{v_i\}_{i \in \mathcal{I}}$. If any intersection of sets in $N(\mathcal{V})$ is contractible, then $N(\mathcal{V})$ is homotopy equivalent to $\bigcup_{i \in \mathcal{I}} v_i$.*

In other words, by serving as an algebraic tool for analyzing the topology of the union $\bigcup_{i \in \mathcal{I}} v_i$, the nerve complex reduces continuous topological question to discrete combinatorial computations, enabling more straightforward calculation of homotopy preservation.

### E. Dominated Vertices and Homotopy Equivalence

A concept of vertex domination provides a useful tool for understanding the topological structure of simplicial complexes and their homotopy equivalence.

**Definition 5** (Dominated vertex)**.** *Consider a simplicial complex $K$ and a vertex $v$, such that $\{v\} \in K$. The vertex $v$ is said to be dominated by another vertex $\hat{v}$ in $K$, if its link $lk(v, K)$ is a simplicial cone $\hat{v}L$, where $L \subset K$.*

When a vertex $v$ is dominated in a simplicial complex $K$, the removal of $v$ does not change the homotopy of $K$ [39], formally expressed as:

$$K - v \simeq K \tag{4}$$

## IV. Path Planning Optimization Formulation

A typical path planning optimization can be formulated as:

$$\min_{\boldsymbol{q}} \sum_{l=1}^{n_q - 1} \|q_{l+1} - q_l\|^2 + o_i(q_{n_1}) + o_g(q_{n_q}) \tag{5a}$$

$$\text{s.t. } g(\boldsymbol{q}) \geq \hat{d} \tag{5b}$$

where $q_l \in \mathbb{R}^{n_r}$ is $l$-th waypoint of the robot, $n_r$ is the dimension of the state of the robot, $n_q$ is the number of the waypoints, $\boldsymbol{q} = \{q_1, \ldots, q_{n_q}\}$ is the sequence of the waypoints of the robot, $o_i(q_{n_1}), o_g(q_{n_q})$ are the cost term for initial pose and desired goal respectively, $g(\boldsymbol{q})$ is the distance between the robot at the waypoints and the environment, and $\hat{d} \in \mathbb{R}^+$ is the safe distance. Notably, the formulation offers flexibility, allowing additional constraints or optimization variables to be incorporated as needed.

We consider obstacles in environment decomposed into a set of convex objects $\mathcal{V}_{tot}$. The environment can be characterized with a nerve complex of these convex objects, denoted as $N(\mathcal{V}_{tot})$, which we refer to as an *environment complex*. In what follows, we refer to each convex object in $\mathcal{V}_{tot}$ as a vertex of the environment complex $N(\mathcal{V}_{tot})$. Note that any nonempty intersection of convex objects forms a contractible space [38]. Consequently, by Theorem 1, $N(\mathcal{V}_{tot})$ and $\mathcal{V}_{tot}$ become homotopy equivalent, providing a foundation for topological analysis of the environment using $N(\mathcal{V}_{tot})$.

Before delving into the details of the methodology, it is important to highlight the key assumptions underlying our framework:

1) The polyhedral mesh representation of the robot is assumed to be known.
2) The SDF for each convex object in $\mathcal{V}_{tot}$ is assumed to be precomputed and available for use.

Under these assumptions, the distance between the robot at the waypoints and the environment, $g(\boldsymbol{q})$ can be calculated as:

$$g(\boldsymbol{q}) = \min_{x \in W(\boldsymbol{q})} \text{sd}_{\mathcal{V}_{tot}}(x)$$

where $W(\boldsymbol{q})$ is the union of the given polyhedral meshes of the robot at the waypoints $q_1, \cdots, q_{n_q}$.

We aim to solve the narrow passage path planning problem (5) using the homotopy optimization method, which starts with simplified constraints and gradually transforms them

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



into the original constraints (5b). The relaxed subproblem, parameterized by the homotopy parameter $\alpha \in [0, 1]$, can be formulated by substituting the collision constraint as:

$$g_k(\boldsymbol{q}, \alpha) \geq \hat{d} \tag{6}$$

where $g(\boldsymbol{q}, \alpha)$ is the homotopy-transformed gap function. This transformed gap function is formulated through an SDF interpolation between convex objects, which is described in detail in Sec. V and Sec. VI.

## V. SDF Interpolation between Convex Objects

To construct the continuous interpolation of the collision avoidance constraint from a simplified constraints (6) to the original complex contraints, we initialize the environment with a subset of the convex objects $\mathcal{V}_0 \subset \mathcal{V}_{tot}$. The gradual transformation from these simplified initial constraints to the original complex constraints is realized by continuously interpolating the constraints as additional convex objects are progressively introduced. As a foundation for this continuous interpolation, we first define and explore the characteristics of SDF interpolation between two intersecting convex objects.

### A. SDF Interpolation between Convex Objects

We first define an *interpolated SDF* between two convex objects $v_1$ and $v_2$ as:

$$\mathrm{sd}_{v_1 \to v_2}(x, \alpha) := (1 - \alpha) f(\mathrm{sd}_{v_1}(x)) + \alpha f(\mathrm{sd}_{v_2}(x)) \tag{7}$$

where $\alpha \in [0, 1]$ is an interpolation variable, $f : \mathbb{R} \to \mathbb{R}$ is a shaping function that satisfies following conditions:

**Property 1** (Shaping function)**.** *A shaping function $f$ satisfies following conditions:*

1) $f(0) = 0$
2) *$f$ is increasing, and convex*

Then, let us define *interpolated object* $v_{v_1 \to v_2}(\alpha)$ from the interpolated SDF as:

$$v^{\alpha}_{v_1 \to v_2} := \mathcal{O}\left(\mathrm{sd}^{\alpha}_{v_1 \to v_2}(\cdot, \alpha)\right) \tag{8}$$

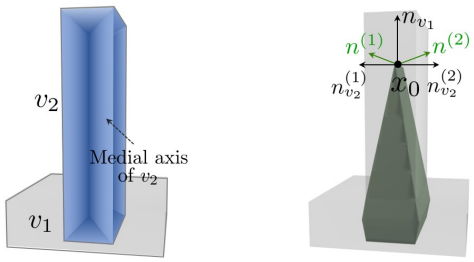
This interpolated object transitions from $v_1$ at $\alpha = 0$ to $v_2$ at $\alpha = 1$. During this transformation, regardless of the shaping function used, the intermediate shapes retain important properties: they remain convex, contain $v_1 \cap v_2$, and contained in $v_1 \cup v_2$, as can be seen in Fig. 4.

**Proposition 1.** *If two convex objects $v_1$ and $v_2$ intersect (i.e., $v_1 \cap v_2 \neq \varnothing$), the interpolated object, denoted as $v_{v_1 \to v_2}(\alpha)$ (8) is convex, and satisfies the following:*

$$v_1 \cap v_2 \subseteq v_{v_1 \to v_2}(\alpha) \subseteq v_1 \cup v_2 \tag{9}$$

*Proof.* Since SDF of a convex object is a convex function [40], both $\mathrm{sd}_{v_1}(\cdot)$ and $\mathrm{sd}_{v_2}(\cdot)$ are convex functions. Considering the properties of the shaping function, which are convex and increasing, $f(\mathrm{sd}_{v_1}(\cdot))$ and $f(\mathrm{sd}_{v_2}(\cdot))$ also retain convexity.

Moreover, the function $\mathrm{sd}_{v_1 \to v_2}(\cdot, \alpha)$, defined as a non-negative linear combination of the convex functions, inherently preserves convexity. Therefore, the interpolated object $v_{v_1 \to v_2}(\alpha)$ forms a convex space, since a sublevel set of a convex function is also a convex [41].




(a) Visualization of medial axis of $v_2$ (b) Sharp ridge of the interpolated object formed on $x_0$

Fig. 3: Given $x_0$ on the medial axis of $v_2$ and outside $v_1$, sharp ridge is generated on $x_0$ for some $\alpha$ when using identity shaping function $f(x) = x$.

For any $x \in \mathbb{R}^3$ that is in $v_1 \cap v_2$, following is satisfied:

$$\begin{aligned} &\mathrm{sd}_{v_1}(x) \leq 0 \text{ and } \mathrm{sd}_{v_2}(x) \leq 0, \\ \Rightarrow\ &\mathrm{sd}_{v_1 \to v_2}(x, \alpha) \leq 0 \end{aligned}$$

Moreover for arbitrary $x \in v_{v_1 \to v_2}(\alpha)$, following is satisfied:

$$\begin{aligned} &\mathrm{sd}_{v_1 \to v_2}(x, \alpha) \leq 0 \\ \Rightarrow\ &\mathrm{sd}_{v_1}(x) \leq 0 \text{ or } \mathrm{sd}_{v_2}(x) \leq 0 \end{aligned}$$

Therefore, $v_1 \cap v_2 \subseteq v_{v_1 \to v_2}(\alpha) \subseteq v_1 \cup v_2$ is satisfied. □

### B. Alleviating Sharpness using Shaping Function

The shaping function $f(\cdot)$ is designed to allieviate the sharpness of the surface of the interpolated object. Without such a shaping function, i.e., $f(x) = x$, the SDF interpolation (7) becomes a linear interpolation. Then, sharp ridges are formed on the medial axis of each object, which is the set of points equidistant from more than one closest point on the surface [37], as illustrated in Fig. 3. These sharp ridges can cause abrupt changes in the contact normals between adjacent waypoints, which is highly undesirable for the optimization.

Consider two intersecting convex objects $v_1$ and $v_2$, and $x_0 \in \mathbb{R}^3$ which lies on the medial axis of $v_2$ and outside $v_1$. We demonstrate that there always exists $\alpha$ such that $v_{v_1 \to v_2}(\alpha)$ has sharp ridge at $x_0$, and the sharpness is alleviated when shaping function is utilized.

*1) Linear interpolation:* We can define a value $\alpha_l$ such that $x_0$ lies on the surface of an interpolated object $v_{v_1 \to v_2}(\alpha_l)$:

$$(1 - \alpha_l)\mathrm{sd}_{v_1}(x_0) + \alpha_l \mathrm{sd}_{v_2}(x_0) = 0 \tag{10}$$

Given $\mathrm{sd}_{v_1}(x_0) > 0$ and $\mathrm{sd}_{v_2}(x_0) < 0$, $\alpha_l \in [0, 1]$ is satisfied.

SDF is not differentiable at points on the medial axis [42]. Since $x_0$ lies on the medial axis of $v_2$, $\mathrm{sd}_{v_2}(x_0)$ is not differentiable at this point. Consequently, there are multiple possible directional derivatives at $x_0$, each pointing to a different closest point on the surface of $v_2$. According to (1), the surface normal of $v_{v_1 \to v_2}(\alpha_l)$ is obtained by differentiating (10). However, this normal cannot be uniquely defined because $\mathrm{sd}_{v_2}(x_0)$ lacks a unique derivative at that point. Instead, we obtain several possible unnormalized normal vectors, such as $n^{(1)}$ and $n^{(2)}$:

$$\begin{aligned} n^{(1)} &:= (1 - \alpha_l) n_{v_1} + \alpha_l n^{(1)}_{v_2} \\ n^{(2)} &:= (1 - \alpha_l) n_{v_1} + \alpha_l n^{(2)}_{v_2} \end{aligned} \tag{11}$$

where $n_{v_1} := \nabla \mathrm{sd}_{v_1}(x)|_{x=x_0}$ is the normal of $v_1$ at $x_0$, and $n_{v_2}^{(1)}$ and $n_{v_2}^{(2)}$ are two different directional derivatives of $\mathrm{sd}_{v_2}(x)$. The discrepancy between $n_{v_2}^{(1)}$ and $n_{v_2}^{(2)}$ leads to a discontinuity in the surface normal of the interpolated object $n^{(1)}$ and $n^{(2)}$, resulting in the formation of a sharp edge, as shown in Fig. 3b.

*2) Shaped interpolation:* To alleviate this sharp ridges, a properly designed shaping function $f(\cdot)$ can be used. Let $\alpha_s$ be a interpolation variable chosen so that $x_0$ lies on the surface of the interpolated object $v_{v_1 \to v_2}(\alpha_s)$ shaped by $f$:

$$(1-\alpha_s) f(\mathrm{sd}_{v_1}(x_0)) + \alpha_s f(\mathrm{sd}_{v_2}(x_0)) = 0 \tag{12}$$

where, given that $f(\mathrm{sd}_{v_1}(x_0)) > 0$ and $f(\mathrm{sd}_{v_2})(x_0) < 0$, $\alpha_s \in [0,1]$ is satisfied. The surface normals at $x_0$ can also be obtained by differentiating (12):

$$\begin{aligned} n^{(1)} &= (1-\alpha_s) f'(\mathrm{sd}_{v_1}(x_0)) n_{v_1} + \alpha_s f'(\mathrm{sd}_{v_2}(x_0)) n_{v_2}^{(1)} \\ n^{(2)} &= (1-\alpha_s) f'(\mathrm{sd}_{v_1}(x_0)) n_{v_1} + \alpha_s f'(\mathrm{sd}_{v_2}(x_0)) n_{v_2}^{(2)} \end{aligned} \tag{13}$$

where $f'$ is the gradient of $f$.

*3) Comparison of sharpness:* To quantify the sharpness of the interpolated object, we define a *discontinuity-inducing normal ratio* $\gamma$. It is a measure of the relative size of the weight of the discontinuous normal in the weighted summation in (11) and (13). Since $n_{v_2}^{(1)}$ and $n_{v_2}^{(2)}$ induces the discontinuity, $\gamma$ for each linear interpolation and shaped interpolation is:

$$\begin{aligned} \gamma_{linear} &= \frac{\alpha_l}{1-\alpha_l} \\ \gamma_{shaped} &= \frac{\alpha_s}{1-\alpha_s} \frac{f'(\mathrm{sd}_{v_2}(x_0))}{f'(\mathrm{sd}_{v_1}(x_0))} \end{aligned} \tag{14}$$

A larger value of $\gamma$ implies a larger weight for the discontinuity-inducing normal, which results in larger angular discrepancy between the surface normals, creating sharper ridges. Following proposition shows that by introducing the shaping function, $\gamma$ is reduced, so that the sharp edges are alleviated.

**Proposition 2.** *With shaping function satisfying Property 1, for normal vectors on a medial axis of $v_2$, it follows that*

$$\gamma_{linear} \geq \gamma_{shaped}$$

*Proof.* See Appendix B. □

One example of the shaping function is modeled using an exponential formula, which adheres to Property 1:

$$f(x) = \frac{1}{\eta}\left(\exp(\eta x) - 1\right) \tag{15}$$

where $\eta \in \mathbb{R}^+$ acts as a scaling factor. It is noteworthy that as $\eta$ approaches zero, (15) converges to $f(x) \approx x$, thereby simplifying the proposed interpolation (7) to linear interpolation between $\mathrm{sd}_{v_1}(\cdot)$ and $\mathrm{sd}_{v_2}(\cdot)$. Fig. 4 visually demonstrates how varying values of $\eta$ affect the shape of the interpolated object. As $\eta$ increases, the interpolated object becomes increasingly smoothed, eliminating the sharp ridges. However, when $\eta$ becomes too large, the exponential shaping function introduces significant nonlinearity into the collision avoidance constraints, deviating from the local linearity of the SDF in the normal direction, and can therefore slow convergence.

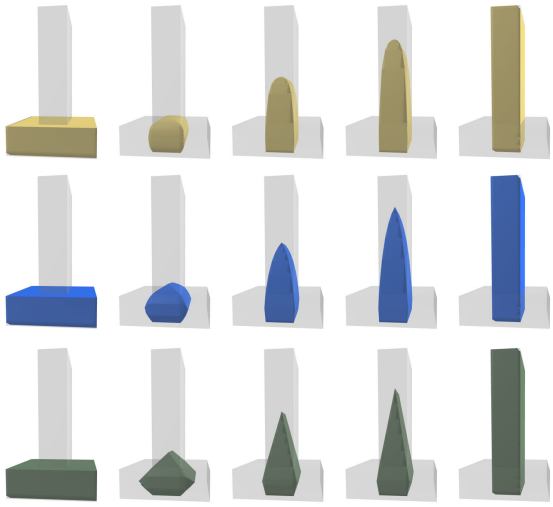


Fig. 4: Shape of the interpolated object using proposed shaping function (15) with $\eta = 40$ (top row), $\eta = 15$ (middle row) and linear interpolation $\eta \to 0$ (bottom row)

## VI. Homotopy-Preserving Collision Constraints Interpolation

In this section, we aim to propose a collision constraints interpolation between the initial simplified constraints to the original complex constraints. This interpolation is constructed by continuously interpolating the constraints while progressively introducing convex objects into the environment, utilizing the SDF interpolation proposed in Sec. V.

While this framework leverages HOM to handle complex constraints, it inherits a known limitation: the transformed problem may disrupt the homotopic landscape, due to emergence of abbreviated paths or folds [14], increasing the risk that the path become stuck in infeasibility. To mitigate these issues, we aim to preserve the homotopy equivalence of the occupied space in Euclidean space. When homotopy equivalence is preserved, the path is more likely to be continuously updated to stay within the free space (Fig. 5b). Conversely, if this homotopy equivalence is lost, the trajectory could stuck in infeasibility or severely torn by obstacles, unable to retain in the free space through any continuous deformation (Fig. 5a). Therefore, we aim to propose a collision constraint interpolation that preserves homotopy of the environment.

### A. Collision Constraint Interpolation using SDF and Collapsible Sets

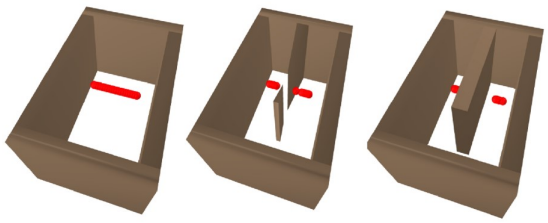
To achieve a homotopy-preserving collision constraints interpolation, we construct a sequence of environments by progressively introducing convex objects into the initial set $\mathcal{V}_0 \subset \mathcal{V}_{tot}$. At each $k$-th sequence, a new set of objects $\mathcal{V}_k^c \subset \mathcal{V}_{tot} \setminus \mathcal{V}_k$ among the remaining objects is introduced into the *current set* of objects $\mathcal{V}_k$:

$$\mathcal{V}_{k+1} = \mathcal{V}_k \cup \mathcal{V}_k^c$$

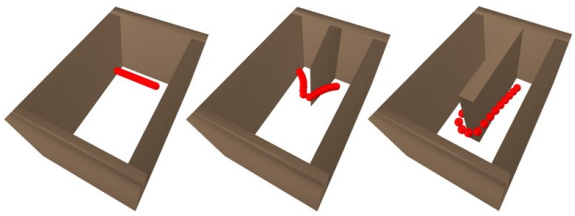
We aim to show that the homotopy-preserving collision constraint interpolation from $\mathcal{V}_k$ to $\mathcal{V}_{k+1}$ can be derived when $\mathcal{V}_k^c$ is a *collapsible set* of $\mathcal{V}_{k+1}$, defined as following.

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



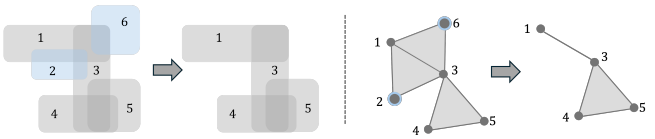


(a) The path cannot stay feasible with continuous updates, due to the change in homotopy.

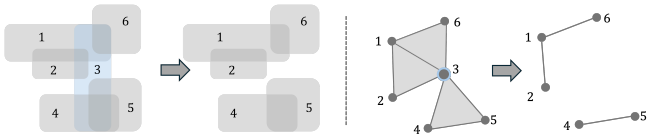


(b) Using our framework, the path can maintain feasibility with continuous update, leading to successful path planning.

Fig. 5: Comparison of homotopy equivalence and the corresponding path planning outcomes when performing constraint interpolation using two distinct formulas.



(a) Valid removal of a collapsible set $\mathcal{V}^c = \{v_2, v_6\}$: both the geometry (left) and the environment complex (right) remain connected, preserving homotopy.



(b) Invalid removal of a vertex $v_3$: the geometry(left) and the complex (right) both split, breaking homotopy.

Fig. 6: Examples of valid and invalid sequneces of removing convex objects (left), and their corresponding environment complexes (right).

**Definition 6** (Collapsible set)**.** *A set of vertices $\mathcal{V}^c$ is called a collapsible set of $\mathcal{V}$ if:*

1) *Each vertex in $\mathcal{V}^c$ is dominated in $N(\mathcal{V})$, by another vertex in $\mathcal{V} \setminus \mathcal{V}^c$.*
2) *No vertices in $\mathcal{V}^c$ intersect with each other.*

Note that removing this collapsible set $\mathcal{V}^c$ from $\mathcal{V}$ preserves homotopy type [39]. Fig. 6 illustrates two examples of removing vertices in an environment complex. In Fig. 6a, the set of vertices $\{v_2, v_6\}$ is a collapsible set (Definition 6) and its removal preserves homotopy. In contrast, in Fig. 6b, removing vertex $v_3$, which is not dominated by any other vertex, tears the complex and breaks homotopy preservance.

Let $\mathcal{V}_k^c = \{v_1^c, \dots, v_{n_c}^c\}$ and $\mathcal{V}_k = \{v_1, \dots, v_n\}$, where $n_c$ and $n$ are the number of vertices in $\mathcal{V}_k^c$ and $\mathcal{V}_k$ respectively. Note that $n_c$ and $n$ depend on $k$, but for brevity, we omit the dependence in the notation. Since $\mathcal{V}_k^c$ is a collapsible set of $\mathcal{V}_k \cup \mathcal{V}_k^c$, each vertex $v_i^c \in \mathcal{V}_k^c$ is dominated by $v_{\sigma(i)} \in \mathcal{V}_k$, where $\sigma : \{1, \dots, n_c\} \to \{1, \dots, n\}$ maps the index of the dominating vertex in $\mathcal{V}_k$.

**Algorithm 1** Identification of collapsible set

1: **Input**: Environment complex $K$, vertex set $\mathcal{V}$
2: **Output**: Collapsible set $\mathcal{V}^c$
3: $\mathcal{V}^c \leftarrow [\,]$
4: **for** $v \in \mathcal{V}$ **do**
5:  **if** $\exists\ v^c \in \mathcal{V}^c$ s.t.$\{v, v^c\} \in K$ **then**
6:  continue
7:  **end if**
8:  **for** $\hat{v} \in \mathcal{V}$ that $\{v, \hat{v}\} \in K$ **do**
9:  **if** $\hat{v}$ dominates $v$ **then**
10:  Append $v$ to $\mathcal{V}^c$
11:  Delete $\hat{v}$ from $\mathcal{V}$
12:  **break**
13:  **end if**
14:  **end for**
15: **end for**

Then, we can formulate the interpolated SDF between $\mathcal{V}_k$ and $\mathcal{V}_{k+1}$ as:

$$\begin{aligned}\mathrm{sd}_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(x, \alpha) := \min(\mathrm{sd}_{\mathcal{V}_k}(x), &\mathrm{sd}_{v_{\sigma(1)} \to v_1^c}(x, \alpha), \dots \\ &\mathrm{sd}_{v_{\sigma(n_c)} \to v_{n_c}^c}(x, \alpha))\end{aligned} \tag{16}$$

which interpolates between $\mathcal{V}_k$ and $\mathcal{V}_{k+1}$ as:

$$\mathrm{sd}_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(x, \alpha) = \begin{cases} \mathrm{sd}_{\mathcal{V}_k}(x), & \text{if } \alpha = 0 \\ \mathrm{sd}_{\mathcal{V}_{k+1}}(x), & \text{if } \alpha = 1 \end{cases}$$

Following theorem shows that the occupied space of the interpolated SDF preserves homotopy equivalence.

**Theorem 2.** *Let $\mathcal{V}_k^c$ be a collapsible set of $\mathcal{V}_{k+1}$, as defined in Definition 6. Then, the occupied space of the interpolated collision constraint* (16)*, denoted as $\mathcal{O}\left(sd_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(\cdot, \alpha)\right)$, is homotopy equivalent regardless of $\alpha$.*

*Proof.* See Appendix C. □

Fig. 13 presents interpolation results across diverse environments. As guaranteed by Theorem 2, Fig. 13 confirms that homotopy of the occupied space is preserved throughout the continuous interpolation.

The collapsible set $\mathcal{V}^c$ of a given set $\mathcal{V}$ can be identified using Algorithm 1. While iterating over $v \in \mathcal{V}$, we first check if $v$ does not collide with any other vertices in $\mathcal{V}^c$ to ensure distinctness as required by the second condition on Definition 6 (Line 5-7). Then, if there is a vertex among the adjacent vertices that dominates $v$, $v$ is identified as the element of a collapsible set, and added into $\mathcal{V}^c$. Note that the dominating vertex $\hat{v}$ should be removed from $\mathcal{V}$ to prevent it from being added to $\mathcal{V}^c$, as dominating vertices should be in $\mathcal{V} \setminus \mathcal{V}^c$ (Line 11).

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



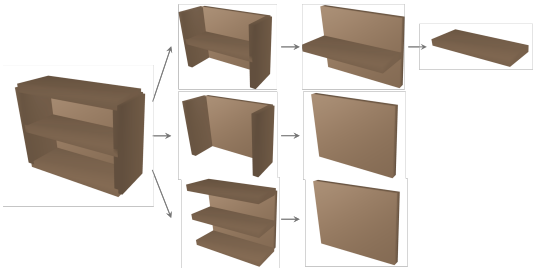


Fig. 7: Three example collapsible set removal sequences on the same environment. Although each sequence removes convex objects in a different order, all three leave a single remaining convex object, visually confirming Thm. 3.

**Algorithm 2** Sequence construction for collapsible sets

1: **Input**: Environment complex $K = N(\mathcal{V}_{tot})$
2: **Output**: Sequence of collapsing sets $V^s$
3: $k \leftarrow 1$
4: **repeat**
5: Find collapsible set $\mathcal{V}_k^c$ (Algorithm 1)
6: $K \leftarrow K - \mathcal{V}_k^c$
7: $k \leftarrow k + 1$
8: **until** $|\mathcal{V}_k^c| > 0$
9: $V^c = \{\mathcal{V}_k^c, \mathcal{V}_{k-1}^c, \ldots, \mathcal{V}_1^c\}$

### B. Sequence Construction for Convex Object Addition

To make the path initialization of the optimization as straightforward as possible, it is desirable to start with a minimal number of initial objects, simplifying the collision constraint. This sequence is constructed by sequentially identifying and removing collapsible sets, starting from the environment complex composed of $\mathcal{V}_{tot}$. The following proposition provides the foundation for our efficient sequence identification algorithm.

**Theorem 3.** *In the environment complex $N(\mathcal{V})$, repeatedly identifying and eliminating collapsible sets, regardless of which collapsible set is chosen at each step, leads to a state where no further collapsible sets remain. At this point, the number of remaining vertices is invariant.*

*Proof.* See Appendix D. □

An example of three distinct sequences for removing collapsible set is shown in Fig. 7. At each step there are multiple choices of which set to remove, so the order of removals can vary. Nonetheless, as Fig. 7 illustrates, regardless of which collapsible set is chosen at each step, the same number of convex objects remains (in this example, a single object). It demonstrates that the environment can always be reduced to its simplest form without exhaustively searching every possible sequence.

The algorithm of identifying the sequence is demonstrated in Algorithm 2. Starting from total convex objects $\mathcal{V}_{tot}$, a collapsible set $\mathcal{V}_k^c$ is identified at each $k$-th step using Algorithm 1. This set $\mathcal{V}_k^c$ is then removed from the environment complex (Line 6), and the process repeats until no further collapsible set can be identified. Finally, the sequence of the collapsible sets is established by reversing the order of $\mathcal{V}_k^c$ (Line 9). The remaining objects, after all collapsible sets have been removed, become the initial objects.

This algorithm is designed to leave as few objects as possible as initial objects. However, if many objects need to be introduced sequentially, the optimization process may take longer. In such cases, the user can choose not to include all possible objects in the collapsible set (Algorithm 1. Line 9 - 13).

## VII. Implementation of Path Planning using HOM

We aim to solve the narrow passage path planning problem (5) using the HOM, which generates a series of subproblems with collision constraints interpolation, as described in Sec. VI. The subproblem corresponding to the $k$-th sequence of collapsible set additions is formulated by substituting the collision constraints (5) as:

$$\min_{\boldsymbol{q}} \sum_{l=1}^{n_q-1} \|q_{l+1} - q_l\|^2 + o_i(q_1) + o_g(q_{n_q}) \quad \text{s.t. } g_k(\boldsymbol{q}, \alpha) \geq \hat{d} \tag{17}$$

where $g_k(\boldsymbol{q}, \alpha)$ is defined using the interpolated SDF (16) as:

$$g_k(\boldsymbol{q}, \alpha) = \min_{x \in W(\boldsymbol{q})} \mathrm{sd}_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(x, \alpha) \tag{18}$$

By sequentially solving these subproblems (17), which involve progressively adding convex objects and gradually increasing the interpolation variable $\alpha$ from 0 to 1 at each sequence, the solution is guided toward the goal. However, several remaining issues arise during this process, which are addressed in detail in this section.

### A. Collision Detection of Interpolated Environment

To solve the path planning optimization (17), collision detection between the robot at the waypoints and the interpolated environment must be performed. However, because the interpolated environment is represented as a signed distance function (SDF) rather than explicit geometry, standard collision detection methods such as Gilbert-Johnson-Keerthi with Expand Polytope Algorithm (GJK-EPA) and broad-phase methods like Axis-Aligned Bounding Box (AABB) or Oriented Bounding Box (OBB), can not be directly applied.

*1) Narrow-phase collision detection:* To accurately compute the minimum distance (i.e. interpolated SDF) between the robot and the interpolated environment, we solve the optimization (18) using the Frank-Wolfe algorithm [43]. Unlike conventional narrow-phase collision detection methods that require access to meshes or support functions, Frank-Wolfe can operate directly on the interpolated SDF by minimizing it over the polyhedral workspace $W(q)$. This avoids the need for costly geometric reconstruction or sampling, making it particularly suitable for our setting where the environment geometry transforms continuously with $\alpha$.

*2) Broad-phase detection:* Broad-phase collision detection also plays a crucial part in path planning by quickly filtering out potential collisions by simplified representations of geometry [44]. To perform broad-phase collision detection,

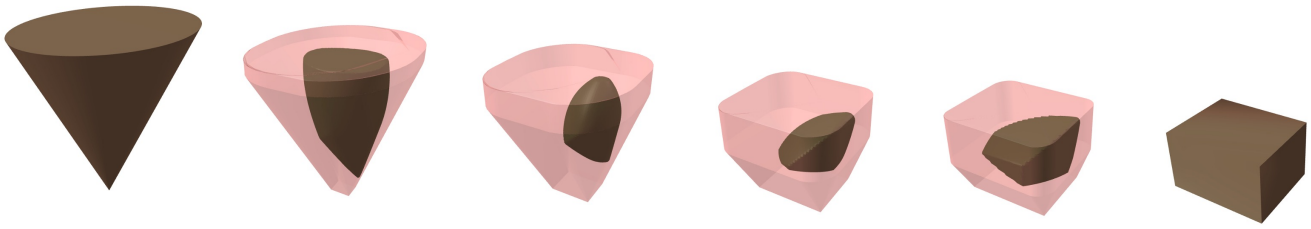


Fig. 8: Process of interpolation between a cone and a box. Occupied space of the interpolated support function (pink) bounds the occupied space of the interpolated SDF (brown).

it is necessary to approximate the occupied space of the interpolated collision constraint (16), which is defined as:

$$\begin{aligned}&\mathcal{O}(\mathrm{sd}_{\mathcal{V}_k\to\mathcal{V}_{k+1}}(\cdot,\alpha))\\&=\mathcal{O}_{\mathcal{V}_k}\cup v_{v_{\sigma(1)}\to v_1^c}(\alpha)\cup\cdots\cup v_{v_{\sigma(n_c)}\to v_{n_c}^c}(\alpha)\end{aligned}\tag{19}$$

Since $\mathcal{V}_k$ is invariant with respect to $\alpha$, its bounding volume, such as AABB or OBB, can be precalculated. In contrast, the geometry of the interpolated objects $v_{v_{\sigma(i)}\to v_i^c}$ varies dynamically with the interpolation variable $\alpha$, making it impossible to precompute their bounding volume in advance. Additionally, extracting their bounding volumes or approximate geometries requires additonal steps like optimization or sampling.

To tackle this, we propose a broad-phase method that generates bounding volumes for the interpolated objects $v_{v_{\sigma(i)}\to v_i^c}(\alpha)$, using support function. Using a support function $h(\cdot):\mathbb{R}^3\to\mathbb{R}$, a corresponding space is represented as:

$$\mathcal{O}^s(h):=\{x\in\mathbb{R}^3\mid y^Tx\le h(y),\quad\forall y\in\mathbb{R}^3\}$$

We define an *interpolated support function* to form a bound of the interpolated objects.

**Definition 7** (Interpolated support function)**.** *Given two support functions $h_{v_1}$ and $h_{v_2}$, we define an interpolated support function as:*

$$h_{v_1\to v_2}(y,\alpha):=(1-\alpha)h_{v_1}(y)+\alpha h_{v_2}(y)\tag{20}$$

Due to the sublinearity of support function, the interpolated support function (20) also retains sublinearity, thereby ensuring that the corresponding space (i.e. $\mathcal{O}^s\left(h_{v_1\to v_2}(\cdot,\alpha)\right)$) forms a unique convex object [45]. Then, following theorem shows that the interpolated object $v_{v_1\to v_2}(\alpha)$ is bounded by the corresponding space of the interpolated support function, as illustrated in Fig. 8.

**Theorem 4.** *The space corresponds to the interpolated support function* (20) *encompasses the $v_{v_1\to v_2}(\alpha)$ represented by an interpolated SDF* (7)*:*

$$v_{v_1\to v_2}(\alpha)\subseteq\mathcal{O}^s(h_{v_1\to v_2}(\cdot,\alpha))$$

*with an arbitrary shaping function that satisfies Property 1.*

*Proof.* See Appendix E. □

The key advantage of utilizing the interpolated support function lies in its convertibility into simple primitives such as AABB or OBB. For example, to calculate AABB of the interpolated object $v_{v_1\to v_2}(\alpha)$, the maximum and minimum extents of the objects along the $x$, $y$ and $z$ axes are required. These values can be derived by evaluating the gradient of the support function in the positive and negative directions along each axis. As an example for the $x$-axis:

$$M_x=\nabla h_{v_1\to v_2}(\vec{x},\alpha),\ \ m_x=\nabla h_{v_1\to v_2}(-\vec{x},\alpha)$$

where $\vec{x}$ is the $x$-axis, $M_x$ and $m_x$ represent the maximum and minimum extents, respectively. Note that this method of using an interpolated support function method provides a tighter bound compared to naively using the union of the meshes of $v_1$ and $v_2$.

Then, the bounding volume for the occupied space (19) can be obtained as the union of the bounding volume of $\mathcal{V}_k$, which is either precomputed or derived from explicit geometry, and the bounding volumes of the interpolated objects $v_{v_{\sigma(i)}\to v_i^c}(\alpha)$, calculated using the interpolated support function (20).

### *B. Adaptive Increase of Interpolation Variable*

The interpolation variable $\alpha$, ranging from 0 to 1, determines the degree of the progression in the interpolation of each sequence. The adjustment of $\alpha$ must balance computational efficiency and collision avoidance. If the increments in $\alpha$ are too small, the process incurs unnecessary computational overhead, especially when the trajectory is far from any collision risk. Conversely, if the increments are too large, the robot may penetrate deeply into the environment, which is undesirable for successful optimization.

To address this issue, we adaptively determine $\alpha$ by finding the maximum value of $\alpha$ that maintains a minimum safe distance $d^*\in\mathbb{R}$, between the robot and the interpolated environment. This is formulated as the following optimization problem:

$$\alpha^*=\underset{\alpha\in[0,1]}{\arg\max}\ \alpha,\ \text{s.t. } g_k(\boldsymbol{q},\alpha)\ge d^*\tag{21}$$

where the constraint can be rewritten from (18) and (16) as:

$$\mathrm{sd}_{\mathcal{V}_k}(x)\ge d^*\tag{22a}$$

$$\mathrm{sd}_{v_{\sigma(i)}\to v_i^c}(x,\alpha)\ge d^*\tag{22b}$$

for all $x\in W(\boldsymbol{q})$ and $i=1,\dots,n_c$.

If the minimum distance between the robot and the current set $\mathcal{V}_k$ falls below $d^*$, the constraint (22a) is violated, rendering the optimization (21) infeasible. In such cases, the optimization is skipped, and $\alpha$ is incremented by a small predefined

**Algorithm 3** Adaptive $\alpha$ determination

1: **Input:** $\mathcal{V}_k$, $\mathcal{V}_k^c$, X
2: Initialize $\alpha^* = 1$
3: **for** each sample $x_j \in X$ **do**
4: **if** $\text{sd}_{\mathcal{V}_k}(x_j) \leq 0$ **then**
5: return $\alpha + \Delta\alpha$
6: **end if**
7: **for** $i = 1, \ldots, n_d$ **do**
8: **if** $\text{sd}_{v_i^c}(x_j) < \text{sd}_{v_{\sigma(i)}}(x_j)$ **then**
9: Calculate $\alpha_{ij}^*$ (24)
10: $\alpha^* \leftarrow \min(\alpha^*, \alpha_{ij}^*)$
11: **end if**
12: **end for**
13: **end for**
14: return $\alpha^*$

value, $\Delta\alpha$. Otherwise, this constraint can be ignored, as it is independent with $\alpha$ and does not influence the solution.

Given the intensive computational demands of the optimization (21) and the relatively low priority of exact optimality, we simplify the process by satisfying the constraint (22b) only at a discrete set of sampled points $X = \{x_1, \ldots, x_{n_x}\} \subset W(\boldsymbol{q})$, where each sampled point $x_j \in \mathbb{R}^3$, rather than over the entire surface $W(\boldsymbol{q})$. The optimization problem is then reformulated as:

$$\begin{aligned} \alpha^* &= \min_{i \in 1:n_c, j \in 1:n_x} \alpha_{ij}^* \\ \alpha_{ij}^* &= \arg\max_{\alpha \in [0,1]} \alpha \text{ s.t. } \text{sd}_{v_{\sigma(i)} \to v_i^c}(x_j, \alpha) \geq d^* \end{aligned} \tag{23}$$

We then solve each low-level problem of (23) for all combinations of $i$ and $j$. For cases of $i$ and $j$ where $\text{sd}_{v_i^c}(x_j) < \text{sd}_{\sigma(i)}(x_j)$, the constraint can be reformulated as:

$$\begin{aligned} & \text{sd}_{v_{\sigma(i)} \to v_i^c}(x_j, \alpha) \geq d^* \\ \Leftrightarrow\ & (1-\alpha) f\left(\text{sd}_{v_{\sigma(i)}}(x_j)\right) + \alpha f\left(\text{sd}_{v_i^c}(x_j)\right) \geq d^* \\ \Leftrightarrow\ & \alpha \leq \underbrace{\frac{d^* - f(\text{sd}_{v_{\sigma(i)}}(x_j))}{f(\text{sd}_{v_i^c}(x_j)) - f(\text{sd}_{v_{\sigma(i)}}(x_j))}}_{\geq 0} \end{aligned}$$

This result defines $\alpha_{ij}^*$, the maximum feasible value of $\alpha$ for this $i$ and $j$:

$$\alpha_{ij}^* = \min\left(1.0,\ \frac{d^* - f(\text{sd}_{v_{\sigma(i)}}(x_j))}{f(\text{sd}_{v_i^c}(x_j)) - f(\text{sd}_{v_{\sigma(i)}}(x_j))}\right) \tag{24}$$

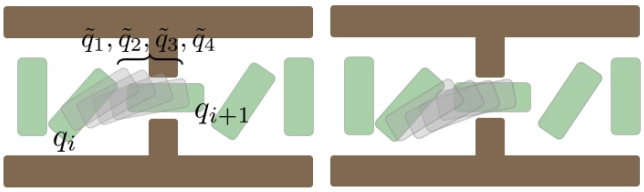
where $0 \leq \alpha_{ij}^* \leq 1$ is satisfied. Note that $\alpha_{ij}^*$ can be efficiently computed since the interpolation is designed to be linear with respect to $\alpha$.

On the other hand, if $\text{sd}_{v_i^c}(x_j) \geq \text{sd}_{v_{\sigma(i)}}(x_j)$, the constraint similarly expands to:

$$\underbrace{\left(f(\text{sd}_{v_i^c}(x_j)) - f(\text{sd}_{v_{\sigma(i)}}(x_j))\right)}_{\geq 0} \alpha \geq \underbrace{d^* - f(\text{sd}_{v_i^c}(x_j))}_{\leq 0}$$

which is always satisfied for $\alpha$ within its valid range $[0, 1]$. As such, this case can be disregarded during optimization.

By combining the cases, the algorithm for adaptive adjustment of $\alpha$ can be derived (Algorithm 3). If there exists a point $x_j$ colliding with the current set $\mathcal{V}_k$, we immediately stop the process and increase $\alpha$ by a small, predefined increments $\Delta\alpha$ (Line 4-6). Otherwise, we calculate an $\alpha^*$ that satisfies (23), by taking the minimum value of $\alpha_{ij}^*$, calculated from (24) (Line 7-12).




(a) Intermediate waypoints initialized with linear interpolation. (b) Optimization result of the intermediate waypoints.

Fig. 9: Refinement process for approximate continuous collision avoidance. The waypoints (green) are obtained from Algorithm 4, and the intermediate waypoints $\tilde{\boldsymbol{q}}$ are initialized through linear interpolation between successive waypoints and subsequently refined through optimization (25) to achieve a smoother, collision-free path.

### *C. Approximate Continuous Collision Avoidance using Post-hoc Path Densification*

Our formulation ensures that collision avoidance constraints are enforced only at the discrete waypoints along the path; however, it does not inherently guarantee collision avoidance between the discrete waypoints. Ensuring continuous collision avoidance is crucial in practice. Yet, analytically enforcing such constraints is generally intractable. As a result, existing methods typically approximate continuous collision avoidance by densifying the path and performing collision checks at interpolated waypoints during the planning process. This approach, while widely adopted, imposes a significant computational burden due to repeated collision detection over many interpolated waypoints.

In contrast, our method decouples the path densification from the planning stage, and handles it as a post-hoc refinement step. This decoupling is made possible by the homotopy-preserving property of the optimization framework, which prevents deep penetration on the intermediate waypoints without explicitly checking it in the planning process (See Fig. 5). Consequently, instead of performing dense collision checking during planning, we can refine each path segment in parallel, yielding a lightweight and scalable post-processing procedure.

Specifically, consider two consecutive waypoints $q_i, q_{i+1}$ obtained from Algorithm 4. Between these we introduce a set of *intermediate waypoints* $\tilde{\boldsymbol{q}} = \{\tilde{q}_1, \ldots, \tilde{q}_{n_{int}}\}$, to provide a finer resolution along the path, where $n_{int}$ specifies the number of interpolated waypoints between $q_i$ and $q_{i+1}$. The refinement of this intermediate waypoints can be formulated as an optimization:

$$\begin{aligned} \min_{\tilde{\boldsymbol{q}}} \|\tilde{q}_1 - q_i\|^2 + \|\tilde{q}_{n_{int}} - q_{i+1}\|^2 + \sum_{l=1}^{n_{int}-1} \|\tilde{q}_{l+1} - \tilde{q}_l\|^2 \\ \text{s.t. } g(\tilde{\boldsymbol{q}}) \geq \hat{d} \end{aligned} \tag{25}$$

**Algorithm 4** Path Planning

**Input:** Convex decomposed obstacle $\mathcal{V}_{tot}$
Construct sequence $V^c$ with Algorithm 2
Initialize $\boldsymbol{q}$ to be feasible in $\mathcal{V}_0$
**for** each sequence $\mathcal{V}_k^c$ in $V^c$ **do**
    $\alpha \leftarrow 0$
    **while** $\alpha < 1$ **do**
        Collision checks (broad+narrow, Sec. VII-A)
        Update $\boldsymbol{q}$ by solving (17)
        Adaptive $\alpha^*$ determination (Sec. VII-B)
        $\alpha \leftarrow \alpha^*$
    **end while**
**end for**
Path refinement for continuous collision avoidance (Sec. VII-C)

Notably, this optimization can be performed independently for each segment defined by consecutive waypoints. Moreover, with the collision-free waypoints already provided, any penetration of intermediate waypoints into obstacles, if present, is minimal. Empirical evidence suggests that this limited penetration can be effectively resolved with a single QP step by linearizing the constraint, achieving satisfactory results. As a result, the optimization process is computationally efficient.

The overall algorithm of solving the path planning (5) through the subproblems (17) is outlined in Algorithm 4. Beginning with an environment that contains only the initial objects $\mathcal{V}_0$, collapsible sets $\mathcal{V}_k^c$ are iteratively added to the current set $\mathcal{V}_k$ in the order determined by Algorithm 2. At each addition, the interpolation variable $\alpha$ is adaptively adjusted from 0 to 1, generating a series of continuously transformed subproblems. Lastly, path refinement is performed for the approximate continuous collision avoidance.

## VIII. Evaluations

We evaluate the performance of our framework using various examples. The proposed algorithm is implemented in Julia [46], and the optimizations (17) and (25) are solved by linearizing the collision constraints and solving the resulting SQP using an efficient ADMM-based solver SubADMM [47].

To determine the success or failure of the planned paths represented by waypoints, we perform a linear interpolation between consecutive waypoints with 50 intervals and check for penetration between the intermediate states and the environment. If none of these intermediate states penetrate the environment, the path is considered successful. The hyperparameters used for each scenarios are detailed in Table I.

For comparison, we include baseline planners from Open Motion Planning Library (OMPL [48]) via MoveIt2 [49] OMPL plugin, TrajOpt using its publicly available C++ implementation, and both STOMP and CHOMP through their MoveIt2 reference implementations, and finally nonlinear programming solver IPOPT [50]. Throughout the comparisons, we report both success time and the total time: the total time represents the statistical measures across all the attempts, while the success time focuses on successful attempts only. When the success rate is low, the mean of the total time approaches the timeout value.

| Method | $\eta$ | $\hat{d}$ | $d^*$ | $n_{traj}$ |
|---|---|---|---|---|
| Dish placing | 1.0 | 0.0025 | -0.002 | 15 |
| Tool extraction | 5.0 | 0.015 | -0.01 | 20 |
| Maze navigation | 3.0 | 0.015 | -0.01 | 20 |
| Humanoid | 5.0 | 0.03 | 0.005 | 20 |

TABLE I: Hyperparameters ($\eta$ for shaping function (15), safe distance $\hat{d}$, adaptive determination threshold $d^*$, and number of waypoints $n_{traj}$) utilized in each scenario.

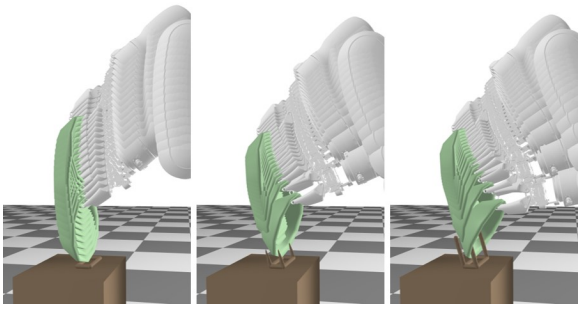


Fig. 10: Snapshots of path optimization process in a dish placing scenario.

### A. Placing Dishes on the Rack

Inserting a dish into a narrow gap of a drying rack is a challenging task for a manipulator. The thinness of the rack makes deep penetration more likely, and in such situations, it becomes difficult to achieve a good contact feature, often resulting in being stuck in infeasibility. For the same reason, even achieving the feasible goal position is challenging for this scenario.

We define the terminal cost as the distance to an approximate reference pose located at the center of the rack. By employing our proposed initialization scheme and refinement process, feasible final position of the plate placed on the drying rack, along with a feasible path could be achieved.

*1) Ablation study:* Table II compares the result of the optimization with and without constraint interpolation. Without constraint interpolation, the path planning optimization (5) is solved using a Sequential Quadratic Programming (SQP), where each QP is solved using SubADMM. Notably, the same method for continuous collision avoidance, as proposed in Sec. VII-C, is applied to ensure a fair comparison between the approaches with and without constraint interpolation. The tests are conducted using combinations of three different dishes and twenty different racks. The results show that our method significantly outperforms the one without constraint interpolation in terms of both success rate and computation time. This highlights the effectiveness of our approach in optimizing goal positions and achieving successful placing paths.

*2) Comparison with baseline planners:* Conventional planning methods typically require the input of a goal pose. To compare our proposed framework with existing methods, we used the feasible final pose obtained by our framework as the goal position for methods other than ours. We conduct tests on various sampling and optimization-based planners in OMPL

| Method | Success | Total time | Success time |
|---|---|---|---|
| Proposed | 55/60 | 1.29 ± 0.28 | 1.27 ± 0.28 |
| Without interpolation | 20/60 | 2.35 ± 0.21 | 2.39 ± 0.26 |

TABLE II: Ablation study of dish placing with and without constraint interpolation.

| **Method** | **Success** | **Total time (s)** | **Success time (s)** |
|---|---|---|---|
| **Proposed** | **18/20** | **1.47 ± 0.36** | **1.43 ± 0.37** |
| RRTConnect | 11/20 | 19.28 ± 9.87 | 11.95 ± 7.38 |
| RRTstar | 0/20 | 30.25 ± 0.38 | - |
| BiTRRT | 12/20 | 17.40 ± 11.88 | 10.32 ± 10.43 |
| TRRT | 2/20 | 27.08 ± 8.60 | 1.29 ± 0.09 |
| BiEST | 9/20 | 18.13 ± 9.89 | 7.80 ± 4.42 |
| BMFT | 3/20 | 27.40 ± 4.12 | 21.72 ± 1.05 |
| PRMstar | 0/20 | 31.60 ± 0.65 | - |
| LazyPRM | 0/20 | 30.04 ± 0.01 | - |
| KPIECE | 4/20 | 22.42 ± 7.69 | 14.36 ± 6.76 |
| BKPIECE | 4/20 | 22.28 ± 6.95 | 14.02 ± 10.55 |
| TrajOpt | 0/20 | 1.26 ± 0.21 | - |
| CHOMP | 2/20 | 22.74 ± 2.58 | 15.28 ± 0.03 |

TABLE III: Comparison with the baseline planners for the task of dish placing, with given goal position achieved from our framework.

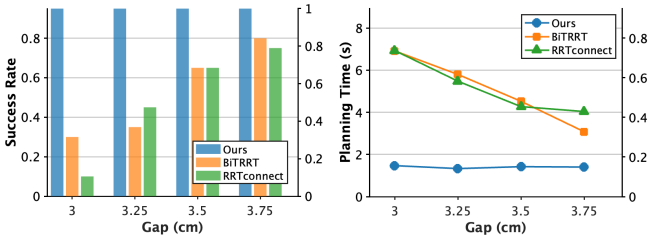



Fig. 11: Scalability of success rate (left) and planning time (right) with respect to the narrowness of the gap. The failure rate of the proposed method is omitted as it successfully completed all tests on 20 different drying racks.

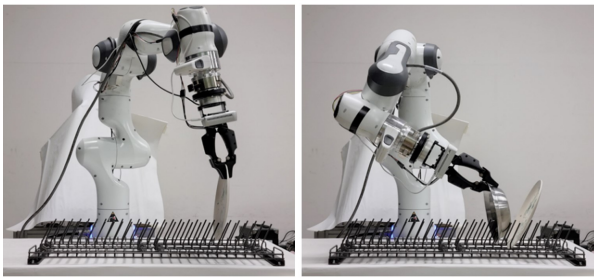


Fig. 12: Results of a successful real-world experiment using the proposed framework for the insertion of two different dishes.

using three different dish shapes and 20 racks. The timeout for the sampling-based methods is set to 30 seconds. The results of these tests are presented in Table III. Despite making the problem easier for the baselines by providing the goal pose, our method still outperformed all others in both success rate and computation time. Particularly, TrajOpt performed poorly in this example, failing the succeed even once. This is likely because TrajOpt uses convex hulls for continuous collision detection, which is inappropriate for highly non-convex shapes like dishes.

*3) Scalability comparison:* Furthermore, Fig. 11 shows the comparison of the scalability with respect to the hardness level of the problem. For the baseline methods, we provide the goal pose computed with our method. For these other methods, we observe that both computation time and success rate deteriorated significantly as the hardness level increased due to changes in sampling efficiency. In contrast, our framework, which is based on optimization and resolves the narrow passage problem through constraint interpolation, maintains nearly consistent success rates and computation times regardless of the hardness level.

*4) Real world experiment:* We also conduct a real-world experiment involving the insertion of dishes, whose geometries are provided in advance as polyderal meshes-into a drying rack using Franka PANDA manipulator. The positions and shapes of the wiry pokes of the drying rack were estimated online [51]. In this procedure, sparse feature points are first tracked across image frames, candidate edges are generated from these points, and their probabilities are updated in a Bayesian framework to recover the rack's wireframe geometry.

Dishes of varying shapes were handed arbitrarily to the robot by a person, and the grasp pose was optimized using interaction data with the ground [52]. Once the grasp pose are available, our framework plans collision-free insertion trajectories, with planning times of 1.23 seconds for the round plate and 1.83 seconds for the bowl, and inserts each dish into the slot that the user specified. This setup enables successful placement of both tested dish shapes, as illustrated in Fig. 12.

### *B. Grasp-Aware Path Optimization for Object Extraction through Gaps*

Manipulating an object in a complex environment is a challenging task for a manipulator, especially when the size of the object is large compared to the gap of the corridors. The success of the task may depend on both the path of the manipulator and how it grasps the object. Existing methods that pre-characterizes the free space in prior [24], [18] face challenges in jointly optimizing the grasp pose and the manipulator path because the configuration of free space depends on the grasp pose.

Specifically, we explicitly parametrize the grasp pose as a variable $\xi \in \mathbb{R}^2$, and reformulate the optimization formulation (5), by introducing a joint optimization variable as $\boldsymbol{z} = (\boldsymbol{q}, \xi)$. The resulting formulation is:

$$\min_{\boldsymbol{z}} \sum_{l=1}^{n_q-1} \|q_{l+1} - q_l\|^2 + o_i(q_{n_1}, \xi) + o_g(q_{n_q}, \xi)$$
$$\text{s.t. } g(\boldsymbol{z}) \geq \hat{d}$$

where $o_i(q_{n_1}, \xi)$ and $o_g(q_{n_q}, \xi)$ represent the initial and goal cost terms, respectively, defined based on the relative pose between the gripper and the object, and $g(\boldsymbol{z})$ encodes the colli-

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



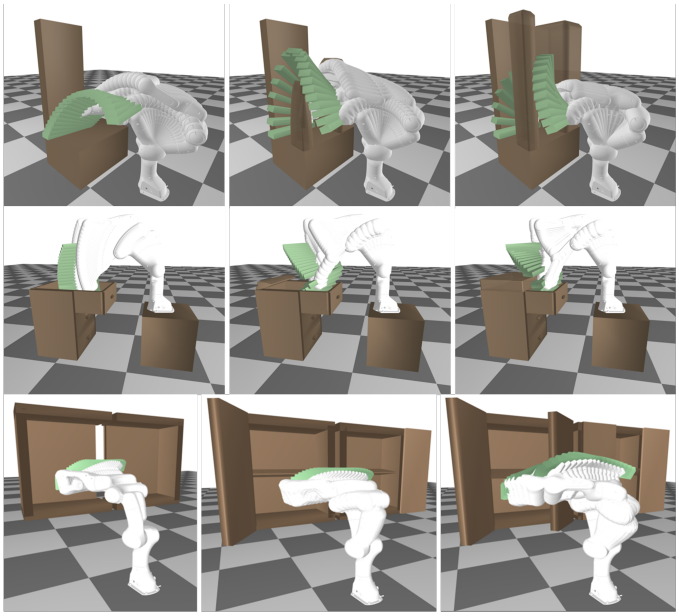


Fig. 13: Snapshots of path optimization process in various environments.

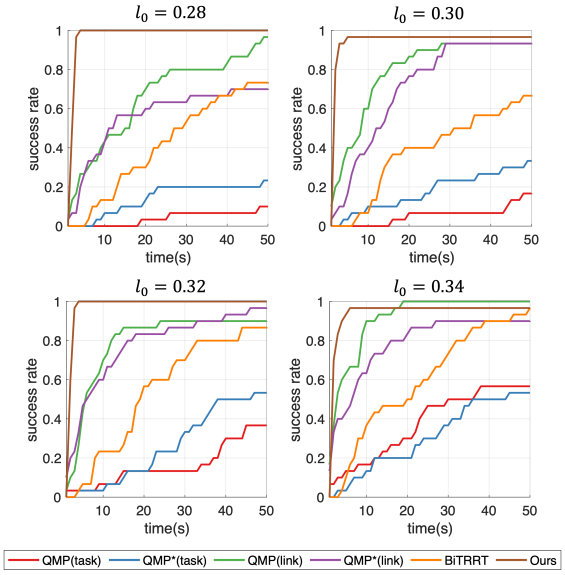



Fig. 14: Comparison of planning performance between the proposed framework, single-level planner (BiTRRT), and multilevel planners (QMP and QMP*) using two hierarchical strategies: link-reduction and task-space hierarchies. Results are shown across four scenarios with varying gap sizes.

sion avoidance constraint, accounting for both the manipulator configuration and the grasp pose.

Fig. 13 shows the result of the optimization for various tasks in different environments. Unlike the method in [15], our formulation can handle more varied environments with complicated connectivity among convex objects, such as mutual connectivity involving three or more convex components (e.g., the middle and the bottom rows of the Fig. 13), which are not supported under the leaf node assumption in [15]. Depending on the configuration, it is observed that the grasp parameters are optimized differently.

*1) Comparison with baseline planners:* To further compare the performance of our framework with other planners, we consider a simplified box extraction scenario through a gap (the top row of the Fig. 13), where the grasp pose is fixed, and only the path of the Franka manipulator is optimized. The scenarios are divided into wide gaps and narrow gaps, where the object length is fixed at 0.4 m. The gap sizes are set to 0.38 m (wide) and 0.28 m (narrow). We introduce slight randomness in the thickness and depth of the obstacles, creating 30 different environments. Each planner is tested under these conditions, with a timeout set to 200 seconds for sampling-based methods.

As seen in Table IV, sampling-based methods achieve high success rates in the wide gap scenario, yet remain an order of magnitude slower than our method. In the narrow gap scenario, the disparity widens further-except for BiTRRT [53], all sampling-based planners record success rates below 50%. The success rate of BiTRRT is marginally higher than ours but incurs an average planning time approximately 22 times longer.

While optimization-based approaches generally perform well in wide gap and achieve comparable performance to our framework, their success rates sharply decline in narrow gap. This is primarily due to difficulties in optimization caused by deep penetration issues. To compare solution quality, we measured path length for successful cases and found that TrajOpt yielded paths that were on average 13% shorter than those generated by our method. This difference arises because TrajOpt and our framework converge to different locally optimal solutions.

*2) Comparison with multilevel planners:* We also compared our result with multilevel planners, specifically the Quotient-space Roadmap Planner (QMP) [54], and its optimal variant QMP* [55], as well as the best-performing single-level planner, BiTRRTs. To evaluate the impact of different multilevel abstractions, we construct two types of hierarchical decompositions. In the first approach, we systematically reduced the number of links considered in the 7-DoF manipulator, defining a three-level hierarchy: the full 7-DoF manipulator, a 5-DoF reduced model, and a minimal 3-DoF model, progressively omitting joints from the end-effector toward the base. In the second approach, we structured the abstraction based on the degrees of freedom of the manipulated tool: the full problem (manipulator and tool), the SE(3) pose of the tool, and the position of the tool in $\mathbb{R}^3$ only. We refer to these two multilevel abstraction schemes as the *link reduction hierarchy* and the *task-space hierarchy*, respectively. As shown in Fig. 14, multilevel planners using the link-reduction hierarchy outperformed the single-level planner, while those using the task-space hierarchy did not, likely due to the overhead of inverse kinematics. Our method consistently achieved faster and more reliable success, especially in narrow-gap scenarios.

*3) Comparison on different shaping function:* We evaluate success rates by varying the parameter $\eta$ in the exponential shaping function (15) for 30 different environments. As shown in Fig. 15 Without smoothing (i.e. $\eta \approx 0$), the success rate was around 0.5. This failure is largely due to inconsistencies in the

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759



| Method | Scenario 1 (Wide gap) | | | Scenario 2 (Narrow gap) | | |
|---|---|---|---|---|---|---|
| | Success | Total time (s) | Success time (s) | Success | Total time (s) | Success time (s) |
| **Proposed** | **30/30** | **1.28 ± 0.16** | **1.28 ± 0.16** | **28/30** | **2.11 ± 0.29** | **2.11 ± 0.29** |
| RRTConnect | 30/30 | 37.57 ± 30.78 | 37.57 ± 30.78 | 8/30 | 198.96 ± 79.15 | 103.51 ± 42.72 |
| RRTstar | 0/30 | 200.11 ± 0.14 | - | 0/30 | 200.25 ± 0.48 | - |
| BiTRRT | 30/30 | 18.89 ± 18.48 | 18.89 ± 18.48 | 29/30 | 50.02 ± 45.49 | 43.98 ± 32.34 |
| TRRT | 0/30 | 200.03 ± 0.00 | - | 0/30 | 200.03 ± 0.00 | - |
| BiEST | 25/30 | 124.52 ± 72.26 | 102.21 ± 51.53 | 2/30 | 196.87 ± 16.43 | 152.46 ± 43.99 |
| BMFT | 26/30 | 118.85 ± 81.58 | 95.47± 56.97 | 11/30 | 220.84 ± 84.17 | 160.98 ± 32.34 |
| PRMstar | 5/30 | 203.41 ± 3.49 | 203.58 ± 2.94 | 4/30 | 201.26 ± 1.23 | 202.87 ± 0.52 |
| LazyPRM | 0/30 | 200.08 ± 0.03 | - | 0/30 | 200.08 ± 0.03 | - |
| KPIECE | 2/30 | 196.46 ± 42.64 | 75.06 ± 41.11 | 0/30 | 197.82 ± 11.99 | - |
| BKPIECE | 25/30 | 132.91 ± 72.74 | 114.56 ± 65.31 | 9/30 | 172.93 ± 88.79 | 59.98 ± 31.30 |
| TrajOpt | 30/30 | 1.69 ± 0.18 | 1.69 ± 0.18 | 17/30 | 3.41 ± 1.09 | 2.71 ± 0.93 |
| CHOMP | 0/30 | 28.17 ± 0.23 | - | 0/30 | - | - |

TABLE IV: Comparison with the baseline planners for the scenario of an object extraction from a narrow gap.

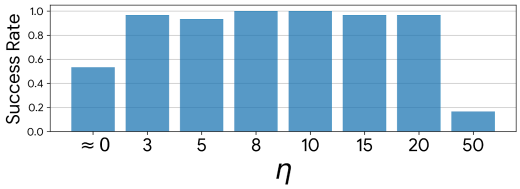



Fig. 15: Success rate with respect to the smoothing parameter $\eta$ in the exponential shaping function (15).

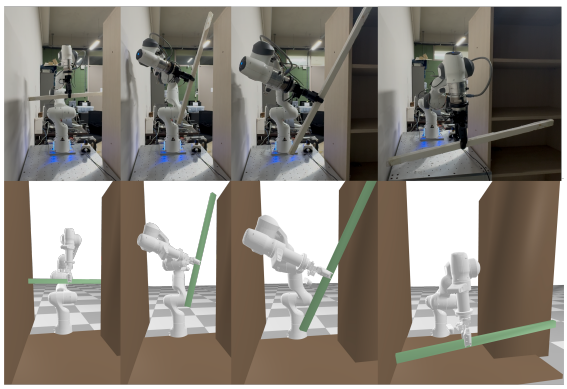


Fig. 16: Real-world execution of a planned trajectory, where Franka successfully transports a box from a narrow gap.

contact features, particularly the contact normals, which cause adjacent waypoints to push in opposite directions, leading to tearing behavior. When $\eta$ was increased to around 50, we observed sharp drop in success rate. This is attributed to a significant increase in the nonlinearity of the interpolated collision constraint, which makes it difficult for the optimizer to converge within a limited number of iterations.

In contrast, over the broad intermediate range from 3 to 20, the success rate remained near optimal (over 93 %), with minimal variation. Although $\eta$ is a tunable hyperparameter, it exhibits a wide, flat optimum, indicating that the framework is robust to the exact choice of $\eta$ in a wide region.

*4) Real world experiment:* We demonstrate the effectiveness of our framework on a real-world task where a 7-DoF Franka robot arm transports a long box through a narrow gap. The gap width is 0.6 m, while the box length is 0.89 m, making the task challenging. Assuming known dimensions and positions of the box and surrounding obstacles, we optimized the path using our framework. The path was computed in 3.8 seconds and was successfully executed on the real robot, as shown in Fig. 16.

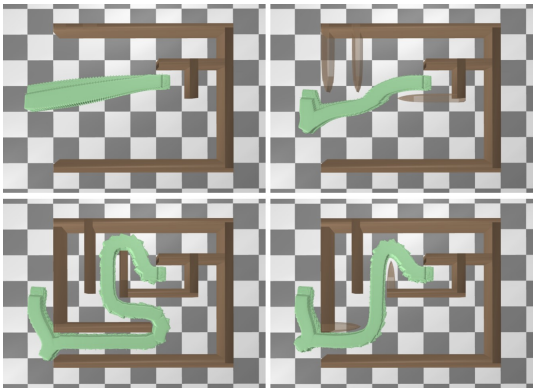


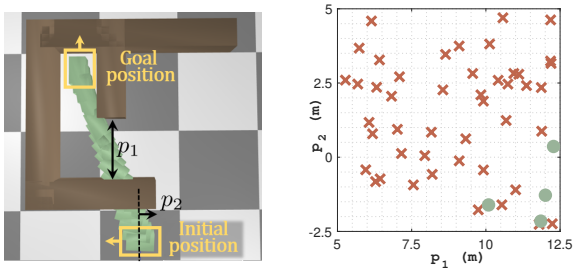
Fig. 17: Process of path planning of four-wheeled vehicle with non-holonomic constraints (clock-wise).

### C. Navigating a Maze with Non-Holonomic Constraints

We also explore the augmentation of additional constraints. Sampling-based path planners generally lack flexibility for constraint augmentation [56], with methods like PRM or RRT requiring constraint enforcement through rejection or projection, both of which involve significant additional computational effort. In contrast, optimization-based path planners, such as our framework, offer greater flexibility in augmenting constraints. We demonstrate this approach in path planning in a maze environment for a vehicle with non-holonomic constraints. We achieve it by augmenting the non-holonomic constraint to the optimization at each subproblem (17). The result of the optimization is illustrated in Fig. 17. Starting from an enlarged initial environment, the path of a vehicle is successfully optimized to navigate through the complex maze environment.

This article has been accepted for publication in IEEE Transactions on Robotics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRO.2025.3641759






(a) Parameters and initial path (b) IPOPT

(c) Without interpolation (d) Proposed method

Fig. 18: Comparison of the success and failure results among IPOPT, the ablation study and our framework in various non-holonomic navigation environments.

To further demonstrate the advantages of our approach in handling complex constraints, we compare our results with those obtained from nonlinear optimization solvers, which solves (5) under non-holonomic constraints. Specifically, we compare our framework against IPOPT and an SQP-based approach. IPOPT is an open-source nonlinear optimization solver that employs an interior-point method. For the SQP-based approach, the optimization is solved by iteratively linearizing the collision constraints and solving the resulting subproblems using SubADMM [47]. Notably, this SQP-based approach corresponds to our framework without constraint interpolation. The environment is randomly generated by controlling parameters $p_1$ and $p_2$, as illustrated in Fig. 18a. The length of the vehicle is set to 5 m.

The results are visualized in Fig. 18. Each $x$ and $y$ axis indicates how the environment is varied. Among 50 trials of random configurations, our method successfully achieved feasible path. On the other hand, IPOPT and SQP tend to fail as the environment complexity increases, particularly when the initial path has deep penetration into the environment, making it challenging to resolve infeasibility. Also, the planning time was shorter in our framework, as can be seen in Table V.

| Method | **Success** | **Total time** | **Success time** |
|---|---|---|---|
| Proposed | 50/50 | 1.40 ± 0.28 | 1.40 ± 0.28 |
| Without interpolation (SQP) | 19/50 | 2.58 ± 0.12 | 2.63 ± 0.096 |
| IPOPT | 4/50 | 3.41 ± 2.02 | 6.91 ± 1.90 |

TABLE V: Comparison of success rate and planning time with IPOPT and ablation case in non-holonomic navigation scenario.

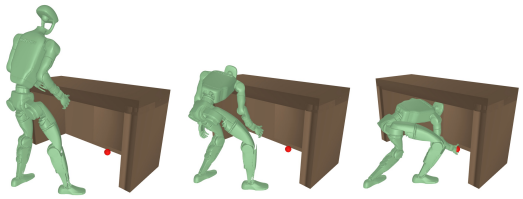


Fig. 19: Optimization results of a humanoid reaching task into the narrow gap of an office desk.

### *D. Humanoid Path Planning Under a Desk*

To demonstrate the applicability of our method to a broader range of systems, we present a proof-of-concept for humanoid path planning, where the task is to reach a target located under an office desk. While humanoid path planning typically requires kinodynamic constraints, we perform kinematic path planning with approximate support polygon constraint by enforcing the center of mass of the torso to remain within an AABB approximation of the support polygon.

The humanoid is modeled as a Unitree G1, with a drawer attached to the desk that further narrows the feasible space. To better navigate the constrained environment, the right foot placement was jointly optimized. We evaluate under ten different desk configurations. The average planning time was 27.09 seconds, and successful plans were generated for all cases, as shown in Fig. 19. Despite the absence of full constraints, we were able to generate plausible paths through kinematic planning alone. In comparison, the ablation study without constraint interpolation succeeded in only one out of ten configurations, and took a higher average planning time of 32.77 seconds.

### *E. Path Variability from Different Sequence of Object Additions*

Our formulation introduces a sequence of convex object additions during the homotopy process, and as discussed in Sec. VI-B, this sequence is non-unique. Different sequences of object addition results different transformations of collision constraints, which guide the optimization toward distinct locally optimal solutions. For instance, in scenarios where the robot can avoid an obstacle by moving to the left, right, or above, the choice of object addition order can guide the optimization toward different paths. An illustrative example is shown in Fig. 20, where multiple different paths are obtained depending on the sequence. While conventional optimization-based path planning methods typically converge to a single locally optimal path, our approach enables diverse exploration by leveraging the non-uniqueness of the homotopy deformation.

## IX. Discussions and Limitations

We propose a path planning framework specifically designed to address the challenges posed by narrow passages. The framework leverages HOM using collision constraints interpolation, ensuring feasibility while maintaining the homotopy equivalence of the free space. By progressively introducing convex objects and interpolating constraints, the method

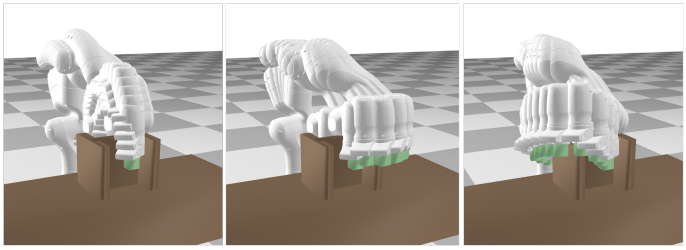


Fig. 20: Results of multiple locally optimal paths obtained from different sequence of additions.

effectively guides the solution toward the goal even in complex environments.

The runtime of our algorithm comprises two primary components. First, the homotopy deformation proceeds over multiple times, where the number reflects the topological complexity of the environment. In highly intricate environment, this interpolation may require a large number of homotopy steps. In the worst case, maintaining topological equivalence throughout the deformation may be infeasible, which represents a limitation of our approach. The second component is the optimization at each subproblem, which we solve via SQP with SubADMM: its per-iteration cost grows nearly linearly in the dimensionality. Therefore, our method exhibits near-linear scaling as the dimensionality of the state space increases.

We would like to address some other limitations of our framework. Firstly, our framework introduces computational overheads such as convex decomposition and environment complex constructions. Moreover, the framework solves the problem by iteratively generating and addressing subproblems, which inherently increases computational time. While our approach is highly efficient for the challenging narrow passage scenarios discussed in this paper, it can be less efficient compared to existing methods for broader passage scenarios, where faster solutions are readily achievable without such overhead.

Also, our algorithm cannot guarantee probabilistic completeness. However, our framework can also be viewed as a methodology for effectively generating collision-free samples for a sampling-based planner. From this perspective, in future work we intend to combine our interpolation module with sampling to guarantee probabilistic completeness.

Furthermore, since the convex objects can only be added under specific conditions, our framework works most effectively when obstacles are represented as convex objects forming relatively simple topologies. In real-world environments, however, nonconvex structures may require complex convex decompositions involving a large number of convex objects, making the representation and path planning more demanding. Moreover, environments reconstructed from sensory data are often incomplete or noisy, which can lead to inaccurate or unnecessarily complex topological structures and further complicate the application of our framework. Nonetheless, such difficulties can be partially mitigated through human intervention during environment construction, such as by approximating nonconvex with a single convex hull to balance representational fidelity and computational tractability, or by manually correcting missing or inconsistent regions in sesory data.

Due to the optimization-based nature of our methodology, there remains the issue of potentially getting stuck in local minima. Also, the sequence of the object addition can influence the specific local minima encountered. However, our framework holds significance in that it increases the likelihood of finding feasible solutions compared to existing optimization-based path planning methods without the proposed framework.

## REFERENCES


[1] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian. Efficient path planning in narrow passages for robots with ellipsoidal components. *IEEE Transactions on Robotics*, 39(1):110–127, 2022.

[2] S. Li and N. T. Dantam. Sample-driven connectivity learning for motion planning in narrow passages. In *IEEE International Conference on Robotics and Automation*, 2023.

[3] A. Orthey and M. Toussaint. Section patterns: Efficiently solving narrow passage problems in multilevel motion planning. *IEEE Transactions on Robotics*, 37(6):1891–1905, 2021.

[4] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

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

[6] L. E. Kavraki, M. N. Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and automation*, 14(1):166–171, 1998.

[7] N. D. Ratliff, M. Zucker, J. A. Bagnell, and S. S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[8] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *IEEE International Conference on Robotics and Automation*, pages 4582–4588, 2011.

[9] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011.

[11] K. Erleben. Methodology for assessing mesh-based contact point methods. *ACM Transactions on Graphics*, 37(3):1–30, 2018.

[12] J. Pan, Z. Chen, and P. Abbeel. Predicting initialization effectiveness for trajectory optimization. In *IEEE International Conference on Robotics and Automation*, pages 5183–5190, 2014.

[13] D. M Dunlavy and D. P O'Leary. Homotopy optimization methods for global optimization. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA ..., 2005.

[14] S. Pardis, M. Chignoli, and S. Kim. Probabilistic homotopy optimization for dynamic motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

[15] M. Lee, J. Lee, and D. Lee. Narrow passage path planning using collision constraint interpolation. In *IEEE International Conference on Robotics and Automation*, 2025.

[16] M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *IEEE International Conference on Robotics and Automation*, pages 2812–2817, 2008.

[17] 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*, 39(5):543–567, 2020.

[18] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *IEEE international conference on robotics and automation*, volume 2, pages 1024–1031, 1999.

[19] L. Zhang and D. Manocha. An efficient retraction-based rrt planner. In *IEEE International Conference on Robotics and Automation*, pages 3743–3750, 2008.

[20] V. Boor, M. H. Overmars, and A. F. Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1018–1023, 1999.

[21] S. Rodriguez, X. Tang, J.-M. Lien, and N. M Amato. An obstacle-based rapidly-exploring random tree. In *IEEE International Conference on Robotics and Automation*, pages 895–900, 2006.

[22] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. In *IEEE International Conference on Robotics and Automation*, pages 7087–7094, 2018.

[23] W. Wang, L. Zuo, and X. Xu. A learning-based multi-rrt approach for robot path planning in narrow passages. *Journal of Intelligent & Robotic Systems*, 90:81–100, 2018.

[24] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake. Shortest paths in graphs of convex sets. *SIAM Journal on Optimization*, 34(1):507–532, 2024.

[25] R. Deits and R. Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pages 109–124. Springer, 2015.

[26] M Yu and C. Fan. Rigid body path planning using mixed-integer linear programming. *IEEE Robotics and Automation Letters*, 2024.

[27] X. Lin, Z. Yang, X. Zhang, and Q. Zhang. Continuation path learning for homotopy optimization. In *International Conference on Machine Learning*, pages 21288–21311. PMLR, 2023.

[28] G. Diaz-Arango, H. Vázquez-Leal, L. Hernandez-Martinez, M. T. S. Pascual, and M. Sandoval-Hernandez. Homotopy path planning for terrestrial robots using spherical algorithm. *IEEE Transactions on Automation Science and Engineering*, 15(2):567–585, 2017.

[29] W. He, Y. Huang, J. Wang, and S. Zeng. Homotopy method for optimal motion planning with homotopy class constraints. *IEEE Control Systems Letters*, 7:1045–1050, 2022.

[30] O B. Bayazit, D. Xie, and N. M Amato. Iterative relaxation of constraints: A framework for improving automated motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3433–3440, 2005.

[31] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, A Yıldırım, U. Filobello-Nino, R. Castaneda-Sheissa, and V. M. Jimenez-Fernandez. Exploring collision-free path planning by using homotopy continuation methods. *Applied Mathematics and Computation*, 219(14):7514–7532, 2013.

[32] C. Von-Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt. Real-time nonlinear shape interpolation. *ACM Transactions on Graphics*, 34(3):1–10, 2015.

[33] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 165–172. 2023.

[34] D. Xu, H. Zhang, Q. Wang, and H. Bao. Poisson shape interpolation. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 267–274, 2005.

[35] A. Kaul and J. Rossignac. Solid-interpolating deformations: construction and animation of pips. *Computers & graphics*, 16(1):107–115, 1992.

[36] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):1–11, 2015.

[37] G. Turk and J. F. O'brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH Courses*, pages 13–es. 2005.

[38] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[39] J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete & Computational Geometry*, 47:301–328, 2012.

[40] S. Yan, X.-C. Tai, J. Liu, and H.-Y. Huang. Convexity shape prior for level set-based image segmentation method. *IEEE Transactions on Image Processing*, 29:7141–7152, 2020.

[41] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[42] R. Wang, Z. Wang, Y. Zhang, S. Chen, S. Xin, C. Tu, and W. Wang. Aligning gradient and hessian for neural signed distance function. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse. Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1):1–17, 2020.

[44] C. Ericson. *Real-time collision detection*. Crc Press, 2004.

[45] R. Schneider. *Convex bodies: the Brunn–Minkowski theory*, volume 151. Cambridge university press, 2013.

[46] T. Koolen and R. Deits. Julia for robotics: Simulation and real-time control in a high-level programming language. In *IEEE International Conference on Robotics and Automation*, pages 604–611, 2019.

[47] J. Lee, M. Lee, and D. Lee. Modular and parallelizable multibody physics simulation via subsystem-based admm. In *IEEE International Conference on Robotics and Automation*, 2023.

[48] I. A Sucan, M. Moll, and L. E Kavraki. The open motion planning library. *Robotics & Automation Magazine*, 19(4):72–82, 2012.

[49] D. Coleman, I. A. Şucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a MoveIt! case study. *Journal of Software Engineering for Robotics*, 5(1):3–16, May 2014.

[50] L. T Biegler and V. M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[51] H. Choi, M. Lee, J. Kang, and D. Lee. Online 3d edge reconstruction of wiry structures from monocular image sequences. *IEEE Robotics and Automation Letters*, 2023.

[52] J. Lee, M. Lee, and D. Lee. Uncertain pose estimation during contact tasks using differentiable contact features. *Robotics: Science and Systems*, 2023.

[53] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

[54] A. Orthey, A. Escande, and E. Yoshida. Quotient-space motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8089–8096, 2018.

[55] A. Orthey, S. Akbar, and M. Toussaint. Multilevel motion planning: A fiber bundle formulation. *The international journal of robotics research*, 43(1):3–33, 2024.

[56] L. Palmieri, S. Koenig, and K. O Arras. Rrt-based nonholonomic motion planning using any-angle path biasing. In *IEEE International Conference on Robotics and Automation*, pages 2775–2781, 2016.

[57] S. An, S. Lee, J. Lee, S. Park, and D. Lee. Collision detection between smooth convex bodies via riemannian optimization framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 12464–12471, 2024.


## APPENDIX A
## SUPPORTING LEMMA

**Lemma 1.** *If and only if $\hat{v}$ dominates $v$ in a complex $K$, following relation satisfies:*

$$st(v, K) \subseteq st(\hat{v}, K)$$

*Proof.* We will prove the statement by demonstrating both directions of equivalence.

*1) $lk(v, K) = \hat{v}L \Rightarrow st(v, K) \subseteq st(\hat{v}, K)$:* Assume that $\mathrm{lk}(v, K) = \hat{v}L$. Then, we have:

$$\mathrm{st}(v, K) = v\mathrm{lk}(v, K) = v\hat{v}L = \hat{v}\hat{L}$$

where $\hat{L} := vL$. Therefore, $\mathrm{st}(v, K)$ is dominated by $\hat{v}$. Then, for an arbitrary face $\sigma \in \mathrm{st}(v, K)$, following holds:

$$\sigma \cup \{\hat{v}\} \in \mathrm{st}(v, K) \subseteq K$$

which indicates that $\sigma \in \mathrm{st}(\hat{v}, K)$. Thus, $\mathrm{st}(v, K) \subseteq \mathrm{st}(\hat{v}, K)$ is satisfied.

*2) $lk(v, K) = \hat{v}L \Leftarrow st(v, K) \subseteq st(\hat{v}, K)$:* Assume that $\mathrm{st}(v, K) \subseteq \mathrm{st}(\hat{v}, K)$ holds. Consider an arbitrary face $\sigma$ that satisfies $\sigma \in \mathrm{lk}(v, K)$. From the definition of link,

$$\sigma \in \mathrm{st}(v, K),\ v \notin \sigma$$

By considering $\tau = \sigma \cup \{v\}$, it follows that $\tau \in \mathrm{st}(v, K)$. From our assumption,

$$\begin{aligned} &\tau \in \mathrm{st}(\hat{v}, K) \\ \Rightarrow\ &\tau \cup \{\hat{v}\} \in K \\ \Rightarrow\ &\sigma \cup \{\hat{v}\} \cup \{v\} \in K \\ \Rightarrow\ &\sigma \cup \{\hat{v}\} \in \mathrm{st}(v, K) \end{aligned}$$

Additionally, since $v \notin \sigma \cup \{\hat{v}\}$, we have $\sigma \cup \{\hat{v}\} \in \mathrm{lk}(v, K)$. Hence, we can deduce that $\mathrm{lk}(v, K) = \hat{v}L$ is satisfied. □

## APPENDIX B
## PROOF OF PROPOSITION. 2

We provide the proof of Proposition 2.

*Proof.* As $x$ lies on the surface of each interpolated object, we have

$$(1-\alpha_l)s_1 + \alpha_l s_2 = 0$$
$$(1-\alpha_s)f(s_1) + \alpha_s f(s_2) = 0$$

Using these conditions and (14), $\gamma$ can be rewritten as:

$$\gamma_{linear} = \frac{\alpha_l}{1-\alpha_l} = -\frac{s_1}{s_2}$$
$$\gamma_{shaped} = \frac{\alpha_s}{1-\alpha_s}\frac{f'(s_2)}{f'(s_1)} = -\frac{f(s_1)f'(s_2)}{f(s_2)f'(s_1)}$$

By applying the convexity of $f$ and $f(0) = 0$ as stated in Property 1, we have:

$$0 \geq f(s) + f'(s)(0-s)$$

for any $s \in \mathbb{R}$. Using this in equality, we have:

$$\begin{aligned}\gamma_{linear} - \gamma_{shaped} &= \frac{f(s_1)f'(s_2)}{f(s_2)f'(s_1)} - \frac{s_1}{s_2} \\ &= \underbrace{\frac{s_1 f'(s_2)}{f(s_2)}}_{\leq 0}\left(\underbrace{\frac{f(s_1)}{s_1 f'(s_1)}}_{\leq 1} - \underbrace{\frac{f(s_2)}{s_2 f'(s_2)}}_{\geq 1}\right) \\ &\geq 0\end{aligned}$$

since $f'(s_1), f'(s_2) > 0$, $f(s_1) > 0, f(s_2) < 0$ from the property of the shaping function and the fact that $s_1 > 0$ and $s_2 < 0$. As a result, $\gamma_{linear} \geq \gamma_{shaped}$ is satisfied. □

## APPENDIX C
## PROOF OF THEOREM 2

We provide the proof of Theorem 2. We begin with a technical lemma that will be used in the proof.

**Lemma 2.** *Assume a vertex $v_i \notin \mathcal{V}$, which is dominated by $\hat{v}_i \in \mathcal{V}$ in $N(\mathcal{V} \cup \{v_i\})$. If $v_i^\alpha := v_{\hat{v}_i \to v_i}(\alpha) \cap v_i$, then $v_i^\alpha$ is dominated by $\hat{v}_i$ in environment complex $K' := N(\mathcal{V} \cup \{v_i^\alpha\})$, i.e.:*

$$st(v_i^\alpha, K') \subseteq st(\hat{v}_i, K')$$

*Proof.* Consider an arbitrary face $\sigma \in \mathrm{st}(v_i^\alpha, K')$. From Proposition 1, we have:

$$\mathcal{O}_\sigma \cap v_i^\alpha \neq \varnothing \;\Rightarrow\; \mathcal{O}_\sigma \cap v_i \neq \varnothing \tag{26}$$

1) Case 1: If $v_i^\alpha \notin \sigma$, following is satisfied from Lemma 1 (Appendix A):

$$\sigma \in \mathrm{st}(v_i, K) \subseteq \mathrm{st}(\hat{v}_i, K) \subseteq \mathrm{st}(\hat{v}_i, K')$$

where $K := N(\mathcal{V})$.

2) Case 2: Otherwise, if $v_i^\alpha \in \sigma$, then let $\tau := \sigma \setminus \{v_i^\alpha\}$, then $\mathcal{O}_\tau \cap v_i \neq \varnothing$ holds from (26).

$$\begin{aligned}&\tau \in \mathrm{st}(v_i, K) \\ \Rightarrow &\{v_i\} \cup \tau \in \mathrm{st}(v_i, K) \subseteq \mathrm{st}(\hat{v}_i, K') \\ \Rightarrow &\mathcal{O}_\tau \cap v_i \cap \hat{v}_i \neq \varnothing\end{aligned}$$

from Lemma 1 (Appendix A). Then, from Proposition 1,

$$\begin{aligned}&\mathcal{O}_\tau \cap v_i^\alpha \cap \hat{v}_i = \mathcal{O}_\tau \cap v_i \cap \hat{v}_i \neq \varnothing \\ \Rightarrow &\sigma \in \mathrm{st}(\hat{v}_i, K')\end{aligned}$$

Therefore, $\mathrm{st}(v_i^\alpha, K') \subseteq \mathrm{st}(\hat{v}_i, K')$ is satisfied. □

We now proceed the proof of Theorem 3.

*Proof.* Geometrically, the occupied space of $\mathrm{sd}_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(\cdot, \alpha)$ can be interpreted as the gluing of interpolated objects into $\mathcal{V}_k$ as:

$$\mathcal{O}\left(\mathrm{sd}_{\mathcal{V}_k \to \mathcal{V}_{k+1}}(\cdot, \alpha)\right) = \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{i=1}^{n_c} v_{v_{\sigma(i)} \to v_i^c}(\alpha) \tag{27}$$

$$= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{i=1}^{n_c} \left(v_{v_{\sigma(i)} \to v_i^c}(\alpha) \cap (v_{\sigma(i)} \cup v_i^c)\right) \tag{28}$$

$$= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{i=1}^{n_c} \left( \underbrace{(v_{v_{\sigma(i)} \to v_i^c}(\alpha) \cap v_{\sigma(i)})}_{\subseteq v_{\sigma(i)} \subseteq \mathcal{O}_{\mathcal{V}_k}} \cup \underbrace{(v_{v_{\sigma(i)} \to v_i^c}(\alpha) \cap v_i^c)}_{:= v_i^\alpha} \right)$$

$$= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{i=1}^{n_c} v_i^\alpha$$

where $v_i^\alpha := v_{v_{\sigma(i)} \to v_i^c}(\alpha) \cap v_i^c$ is a convex object since intersection between two convex objects is convex. The equation (27) can be derived from (2), and the transformation (28) is derived from the property (9), i.e., $v_{v_{\sigma(i)} \to v_i^c}(\alpha) \subseteq v_{\sigma(i)} \cup v_i^c$.

For arbitrary $i = 1, \ldots, n_c$, let $K_{1:i} := N(\mathcal{V}_k \cup \{v_1^\alpha, \ldots, v_i^\alpha\})$. From the distinctness of $\mathcal{V}_k^c$ from the definition of collapsible set, $v_i^\alpha$ not intersect any other $v_j^\alpha$ for arbitrary $j \neq i$:

$$v_i^\alpha \cap v_j^\alpha \subseteq v_i^c \cap v_j^c = \varnothing,\ i \neq j$$

It follows that $v_j^\alpha \notin \mathrm{st}(v_i^\alpha, K_{1:i})$. This implies

$$\mathrm{st}(v_i^\alpha, K_{1:i}) = \mathrm{st}(v_i^\alpha, K_i) \subseteq \mathrm{st}(v_{\sigma(i)}, K_i) \subseteq \mathrm{st}(v_{\sigma(i)}, K_{1:i})$$

from Lemma 2. Therefore, each $v_i^\alpha$ is dominated by $v_{\sigma(i)}$ in $K_{1:i}$, therefore $K_{1:i}$ and $K_{1:i} - v_i^\alpha$ are homotopy equivalent:

$$K_{1:i} \simeq K_{1:i-1}$$

By collapsing each $v_i^\alpha$ in $K_{1:i}$, we obtain the sequence of homotopy equivalences:

$$K_{1:n_c} \simeq K_{1:n_c-1} \simeq \ldots \simeq N(\mathcal{V}_k)$$

Therefore, independent of $\alpha$, the occupied space of the interpolation defined by (16) is homotopic to $\mathcal{V}_k$, implying that the interpolation process preserves homotopy. □

## APPENDIX D
## PROOF OF THEOREM 3

The following provides the proof of Theorem 3.

*Proof.* Let us denote $K \searrow L$ when a complex $K$ can be transformed into another complex $L$ through a sequence

of steps, each eliminating a single dominated vertex, while preserving homotopy equivalence.

Firstly, we aim to show that eliminating a collapsible set $\mathcal{V}^c = \{v_1^c, \cdots, v_{n_c}^c\}$ of a simplicial complex $K$ from $K$ satisfies $K \searrow K - \mathcal{V}^c$. Assume each $v_i^c \in \mathcal{V}^c$ is dominated by $v_{\sigma(i)} \in \mathcal{V}$. Let $K' := K - v_1^c$. After removing $v_1^c$, the following condition is satisfied:

$$st(v_j^c, K') \subseteq st(v_{\sigma(j)}, K')$$

for all $j = 2, \ldots, n_c$. This implies that $\mathcal{V}^c \setminus \{v_1^c\}$ is again a collapsible set of $K'$. By repeating this process, we can iteratively remove all vertices in $\mathcal{V}^c$, yielding:

$$K \searrow K - v_1^c \searrow \cdots \searrow K - \mathcal{V}^c \tag{29}$$

Let $K_0$ denote the remaining complex after eliminating all possible collapsible sets from $K$. By iteratively applying the same process (29), we have $K \searrow K_0$. As shown in [39], $K_0$ is unique. Therefore, the number of vertices after eliminating all possible collapsible set is uniquely determined. □

## APPENDIX E
## PROOF OF THM. 4

We include here the detailed proof of Theorem 4.

*Proof.* Let SDF of a point $x \in \mathbb{R}^3$ to $v_1$ and $v_2$ as $s_1 := \mathrm{sd}_{v_1}(x)$ and $s_2 := \mathrm{sd}_{v_2}(x)$. Also, $h_{v_1}(x)$ and $h_{v_2}(x)$ be the corresponding support functions of $v_1$ and $v_2$ respectively. Then SDF functions (i.e. $s_i(x),\ i = 1, 2$) can be expressed with the support functions as [57]:

$$s_i(x) = \sup_{\|y\|=1} (y^T x - h_{v_i}(y)),\ i = 1, 2$$

From the Property 1, following is satisfied:

$$\begin{aligned} f(s_i(x)) &= f\left(\sup_{\|y\|=1} (y^T x - h_{v_i}(y))\right) \\ &= \sup_{\|y\|=1} f\left(y^T x - h_{v_i}(y)\right) \end{aligned}$$

Assume $x \in v_{v_1 \to v_2}(\alpha)$. Then, following equations hold:

$$\begin{aligned} 0 \geq& (1-\alpha) f\left(s_1(x)\right) + \alpha f\left(s_2(x)\right) \\ =& (1-\alpha) \sup_{\|y\|=1} f(y^T x - h_{v_1}(y)) + \alpha \sup_{\|y\|=1} f(y^T x - h_{v_2}(y)) \\ \geq& \sup_{\|y\|=1} \left\{(1-\alpha) f(y^T x - h_{v_1}(y)) + \alpha f(y^T x - h_{v_2}(y))\right\} \\ \geq& \sup_{\|y\|=1} \left\{f\left((1-\alpha)(y^T x - h_{v_1}(y)) + \alpha(y^T x - h_{v_2}(y))\right)\right\} \\ \geq& \sup_{\|y\|=1} \left\{f\left(y^T x - ((1-\alpha) h_{v_1}(y) + \alpha h_{v_2}(y))\right)\right\} \end{aligned}$$

This leads to the inequality for all $\|y\| = 1$:

$$\begin{aligned} & f\left(y^T x - ((1-\alpha) h_{v_1}(y) + \alpha h_{v_2}(y))\right) \leq 0 \\ \Rightarrow\ & y^T x \leq h_{v_1 \to v_2}(y, \alpha) \end{aligned} \tag{30}$$

As the left and right term are both linear to $y$, (30) is satisfied for all $y \in \mathbb{R}^3$, therefore the point $x$ lies within the space represented by the interpolated support function (20). Consequently, the space represented by interpolated SDF is a subset of the space represented by interpolated support function. □

**Minji Lee** received the B.S. degree in Mechanical Engineering in 2019 from Seoul National University, Seoul, Republic of Korea, and the Ph. D. degree in Mechanical Engineering from the same institute in 2025. She is currently a Senior Research Engineer at Holiday Robotics. Her main research interests include motion planning, contact modeling and manipulation with multi-contact.

**Jeongmin Lee** received the B.S. degree in Mechanical Engineering from Seoul National University, Seoul, Republic of Korea, in 2019, and the Ph.D. degree in Mechanical Engineering from the same institution in 2024. He is currently a Senior Research Engineer at Holiday Robotics. His main research interests include optimization, physics simulation, and multi-contact robotic manipulation.

**Dongjun Lee** received his B.S. degree in mechanical engineering and an M.S. degree in automation and design from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, and a Ph.D. degree in mechanical engineering from the University of Minnesota at Twin Cities, Minneapolis, MN, USA, 2004. He is currently a Professor with the Department of Mechanical Engineering, Seoul National University, Seoul, South Korea. His main research interests are the dynamics and control of robotic and mechatronic systems with emphasis on aerial/mobile robots, teleoperation/haptics, physics simulation, multirobot systems, and industrial control applications. Dr. Lee received the U.S. NSF CAREER Award 2009, the Best Paper Awards from IAS 2012 and KROC 2020, and the SNU Excellence in Research Award 2023. He is a Senior Editor of International Journal of Robotics Research and was on the Editorial Boards for IEEE Transactions on Robotics, IEEE Robotics and Automation Letters and IEEE Transactions on Haptics.