

Narrow Passage Path Planning using 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. Given that deep penetration into the environment can cause optimization failure, we propose a framework to ensure feasibility throughout the process using a series of subproblems tailored for narrow passage problem. We begin by decomposing the environment into convex objects and initializing collision constraints with a subset of these objects. By continuously interpolating the collision constraints through the process of sequentially introducing remaining objects, our proposed framework generates subproblems that guide the optimization toward solving the narrow passage problem. Several examples are presented to demonstrate how the proposed framework addresses narrow passage path planning problems.

I. INTRODUCTION

Path planning is a fundamental and crucial aspect of robotic tasks. Path planning problem in narrow passage or cluttered environment remains particularly challenging with active researches having been conducted even to this day ([1], [2], [3], [4]). Such environments are not merely common in industrial settings, but also frequently encountered in household 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) [5] or probabilistic roadmap (PRM) [6]. While these methods offer certain advantages, such as convenient problem formulation and probabilistical completeness [7], they struggle in narrow passages, due to sampling inefficiencies [2].

More recently, optimization-based path planning has emerged as another promising approach ([8], [9], [10], [11]), formulating the path planning problem as an optimization. These methods are able to quickly converge to paths with low cost in terms of path length, smoothness, or other task-specific metrics, by leveraging gradient information and, in some cases, second-order information. They also offer flexibility in integrating various cost and constraint factors. Despite their advantages, as the problem is inherently non-convex when collision avoidance is considered, they can get stuck in local minima and fail to find an optimal or even feasible path without a sufficiently good initialization [10].

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

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.

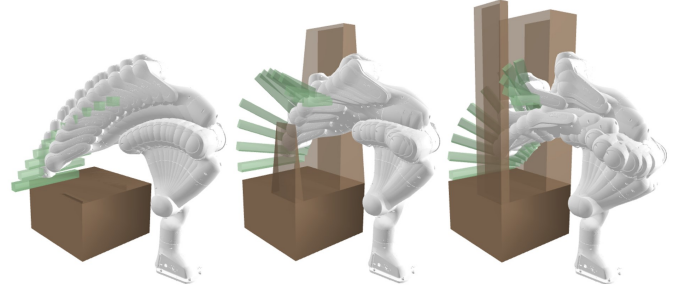


Fig. 1: Optimization results for manipulator path planning during tool extraction from a narrow gap, utilizing the proposed collision constraint interpolation framework.

This difficulty is particularly pronounced in narrow passages, because deep penetration is more likely to occur. There are two major factors that make the deep penetration problematic for trajectory optimization. First, contact features such as witness points, penetration depth, and contact normal are well-defined only when there is either no contact or minimal penetration [12]. Secondly, in situations of deep penetration, especially when the path passes through the medial axis [13], the contact normals between adjacent waypoints may become inconsistent, thus they may push the solutions in opposite directions, making it difficult to escape from infeasibility [10]. In essence, within these narrow passages, traditional path planning methods often fall short, struggling to find feasible paths or consuming excessive computational resources.

In this paper, we propose addressing the problem through a novel optimization-based method with a series of subproblems tailored for narrow passage problems. We first relax the collision constraints so that we can initially start from an expanded free space, and, as the subproblems proceed, the constraints are gradually tightened back to their original form (See Fig. 1). This enables continuous refinement of the path, guiding it toward a feasible solution of the narrow passage problem while preventing deep penetration issues.

More specifically, we start by decomposing the environment into convex objects. Starting with a few initial objects, the remaining objects are then progressively introduced and augmented into the environment. The addition is carried out through a certain interpolation between the Signed Distance Functions (SDFs) of newly-added and existing objects, in such a way that the path is continuously morphed into the final solution of the narrow passage problem without any tearing of the path, by considering the homotopy.

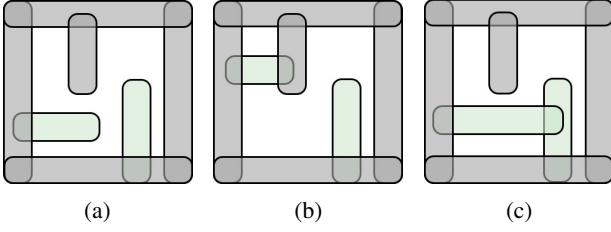


Fig. 2: Given an environment with convex objects \mathcal{V} colored in gray, a set of green convex objects forms leaf set in (a), and not in (b) or (c). In (b), one of the green object violates Condition 1.1 by intersecting with two objects in \mathcal{V} . In (c), the green objects violate Condition 1.2 by intersecting with each other.

The paper is structured as follows: In Sec. II, we introduce our collision constraint interpolation framework. Sec. III details the optimization-based path planning algorithm using this collision constraint interpolation. Path planning examples and comparative analyses are presented in Sec. IV, followed by conclusions in Sec. V.

II. COLLISION CONSTRAINT INTERPOLATION

A key insight of our paper is that maintaining feasibility throughout optimization process is crucial for preventing the path from getting stuck in local minima, as it can avoid deep penetration, which is problematic for the optimization. To achieve this, we propose a series of subproblems that initially simplify the collision constraints to create a wide free space, and then gradually revert it to the original collision constraint as the subproblems progress, while guiding the path to the solution of narrow passage problem.

To implement this framework, we first decompose the environment into a collection of convex objects $\mathcal{V}_{tot} = \{v_1, \dots, v_{n_t}\}$, where n_t is the number of the objects in \mathcal{V}_{tot} . Each object v_i represents the space it occupies in the environment. The environment is initialized with a subset of these objects $\mathcal{V}_{init} \subseteq \mathcal{V}_{tot}$. Each subproblem is defined by gradually interpolating collision constraints of the sequence as new objects are progressively introduced into the environment. At each sequence, a leaf set is introduced, satisfying following condition.

Condition 1 (Leaf set): Let \mathcal{V}^l be a set of convex objects with respect to a connected set of convex objects \mathcal{V} . The set \mathcal{V}^l is a leaf set if it meets the following condition:

- 1) Each element of \mathcal{V}^l intersects with exactly one object from \mathcal{V}
- 2) No elements in \mathcal{V}^l intersect with each other.

Fig. 2 illustrates a leaf set (Fig. 2a) and counterexamples (Fig. 2b, Fig. 2c). Note that the gluing of the leaf set neither creates nor removes cycles, thereby preserving the homotopy of the environment.

To achieve interpolated collision constraints during the sequential addition of the leaf sets, we first examine the properties of SDF. Based on these properties, we define SDF interpolation between two convex objects, and propose an interpolation scheme to be applied across the sequences.

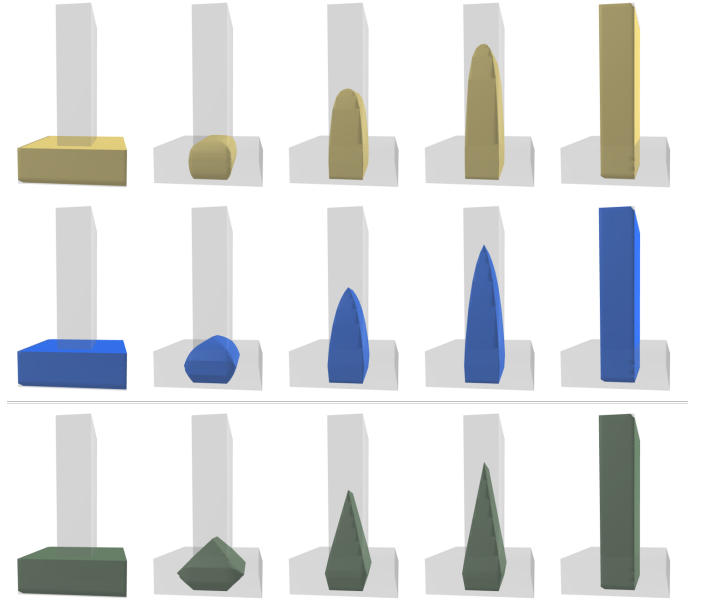


Fig. 3: Object interpolation process using proposed shaping function (5) with $\eta = 40$ (top row), $\eta = 15$ (middle row) and linear interpolation $\eta \rightarrow 0$ (bottom row).

A. Signed Distance Function

A signed distance function (SDF) for an object v is a function that quantifies the distance from any point $x \in \mathbb{R}^3$ to the surface of an object, formally defined as:

$$\text{SDF}_v(x) = \begin{cases} -\inf_{y \in \partial v} d(x-y) & \text{if } x \in v \\ \inf_{y \in \partial v} d(x-y) & \text{else} \end{cases}$$

where ∂v is the boundary of v , and the metric d is the commonly used Euclidean distance. For a collection of objects $\mathcal{V} = \{v_1, \dots, v_n\}$, the combined SDF can be represented as:

$$\text{SDF}_{\mathcal{V}}(x) := \min(\text{SDF}_{v_1}(x), \dots, \text{SDF}_{v_n}(x))$$

Let us define an *occupied space* of a function $g(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}$ as $\mathcal{O}(g) = \{x \in \mathbb{R}^3 \mid g(x) \leq 0\}$. It follows that the occupied space of a SDF of a set of objects, denoted as $\mathcal{O}_{\mathcal{V}}$, is exactly itself:

$$\mathcal{O}_{\mathcal{V}} := \mathcal{O}(\text{SDF}_{\mathcal{V}}) = \bigcup_{v \in \mathcal{V}} v$$

Also, following is satisfied for any functions $g_1(\cdot), g_2(\cdot) \in \mathbb{R}^3 \rightarrow \mathbb{R}$:

$$\mathcal{O}(\min(g_1, g_2)) = \mathcal{O}(g_1) \cup \mathcal{O}(g_2) \quad (1)$$

B. Smoothed SDF Interpolation between Convex Objects

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

$$\text{SDF}_{v_1 \rightarrow v_2}^{\alpha}(x) := (1-\alpha)f(\text{SDF}_{v_1}(x)) + \alpha f(\text{SDF}_{v_2}(x)) \quad (2)$$

where $\alpha \in [0, 1]$ is an interpolation variable and $f : \mathbb{R} \rightarrow \mathbb{R}$ is a shaping function that satisfies the following condition.

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

- 1) $f(0) = 0$
- 2) f is non-decreasing, and convex

Let us define an *interpolated object*, $v_{v_1 \rightarrow v_2}^\alpha$, such that,

$$v_{v_1 \rightarrow v_2}^\alpha := \mathcal{O} \left(\text{SDF}_{v_1 \rightarrow v_2}^\alpha \right) \quad (3)$$

The following proposition outlines the properties of the interpolated object derived from any shaping function that satisfies Property 1.

Proposition 1: If two convex objects v_1 and v_2 intersect (i.e., $v_1 \cap v_2 \neq \emptyset$), their interpolated object $v_{v_1 \rightarrow v_2}^\alpha$ as defined by (3) is convex and satisfies:

$$v_1 \cap v_2 \subseteq v_{v_1 \rightarrow v_2}^\alpha \subseteq v_1 \cup v_2 \quad (4)$$

Proof: Since SDF of convex object is convex [14], both $\text{SDF}_{v_1}(\cdot)$ and $\text{SDF}_{v_2}(\cdot)$ are convex functions. Considering the properties of the shaping function, which are convex and non-decreasing, $f(\text{SDF}_{v_1}(\cdot))$ and $f(\text{SDF}_{v_2}(\cdot))$ also retain convexity.

Moreover, the function $\text{SDF}_{v_1 \rightarrow v_2}^\alpha(\cdot)$, defined as a non-negative linear combination of the convex functions, inherently retains convexity. Given that the level sets of a convex function are convex, it follows that the occupied space of $\text{SDF}_{v_1 \rightarrow v_2}^\alpha$ is convex. Consequently, the interpolated object $v_{v_1 \rightarrow v_2}^\alpha$ is also convex.

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

$$\begin{aligned} \text{SDF}_{v_1}(x) &\leq 0 \text{ and } \text{SDF}_{v_2}(x) \leq 0, \\ \Rightarrow \text{SDF}_{v_1 \rightarrow v_2}^\alpha(x) &\leq 0 \end{aligned}$$

Moreover for arbitrary $x \in v_{v_1 \rightarrow v_2}^\alpha$, following is satisfied:

$$\begin{aligned} \text{SDF}_{v_1 \rightarrow v_2}^\alpha(x) &\leq 0 \\ \Rightarrow \text{SDF}_{v_1}(x) &\leq 0 \text{ or } \text{SDF}_{v_2}(x) \leq 0 \end{aligned}$$

Therefore, $v_1 \cap v_2 \subseteq v_{v_1 \rightarrow v_2}^\alpha \subseteq v_1 \cup v_2$ is satisfied. \blacksquare

The shaping function $f(\cdot)$ is designed to smooth the surface of the interpolated object. For instance, if we do not use this shaping function (i.e. $f(x) = x$), some sharp ridges would be formed, which correspond to the medial axis [15] (See the bottom row of Fig. 3). These sharp ridges can cause abrupt changes in the contact normals between adjacent waypoints, which is highly undesirable for the optimization.

Such sharp ridges can be mitigated by using a properly-designed shape function $f(\cdot)$. One example is modeled using an exponential formula, which adheres to Property 1:

$$f(x) = \frac{1}{\eta} (\exp(\eta x) - 1) \quad (5)$$

where $\eta \in \mathbb{R}^+$ acts as a scaling factor. Fig. 3 visually demonstrates how varying values of η affect the shape of the interpolated object. As η increases, the interpolated object becomes increasingly smoothed, eliminating the sharp ridges.

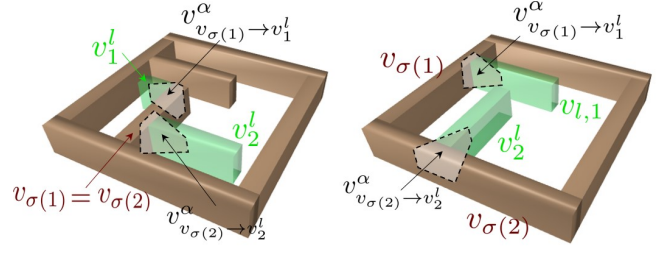


Fig. 4: Illustration of two sequential processes of gluing leaf sets to the environment.

C. Homotopy Preserving Collision Constraint Interpolation

Consider a k -th sequence of adding a leaf set $\mathcal{V}_k^l = \{v_1^l, \dots, v_{n_l}^l\}$, where n_l is the number of objects in the leaf set, into a current environment characterized by objects $\mathcal{V}_k = \{v_1, \dots, v_n\} \subseteq \mathcal{V}_{tot}$, where n is the number of objects in current environment. Note that n and n_l vary with the sequence k , but for brevity, we omit this dependence in the notation. Each convex object v_j^l intersects uniquely with an object $v_{\sigma(j)} \in \mathcal{V}_k$, where $\sigma(\cdot)$ maps the index of the intersecting object in \mathcal{V}_k . See Fig. 4 for the graphical illustrations. Then, this sequence can be interpolated using the interpolation variable α :

$$\text{SDF}_{\mathcal{V}_k + \mathcal{V}_k^l}^\alpha(x) := \min \left(\text{SDF}_{\mathcal{V}_k}(x), \text{SDF}_1^\alpha(x), \dots, \text{SDF}_{n_l}^\alpha(x) \right) \quad (6)$$

where $\text{SDF}_j^\alpha(x) := \text{SDF}_{v_{\sigma(j)} \rightarrow v_j^l}^\alpha(x)$. When $\alpha = 0$, the occupied space is exactly $\mathcal{O}_{\mathcal{V}_k}$, and at $\alpha = 1$, it expands to $\mathcal{O}_{\mathcal{V}_k \cup \mathcal{V}_k^l}$:

$$\text{SDF}_{\mathcal{V}_k + \mathcal{V}_k^l}^\alpha = \begin{cases} \text{SDF}_{\mathcal{V}_k} & \text{if } \alpha = 0 \\ \text{SDF}_{\mathcal{V}_k \cup \mathcal{V}_k^l} & \text{if } \alpha = 1 \end{cases}$$

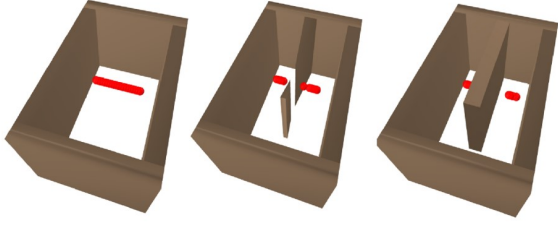
As shown in Fig. 5, preserving homotopy of the occupied space is essential during this interpolation; if lost, the path may stuck in an infeasibility, unable to retain within the free space through any continuous deformation (Fig. 5a). Conversely, if homotopy is preserved, the path is more likely to be continuously updated to stay within the free space (Fig. 5b).

From (1), the occupied space of the interpolated collision constraint (6) can be characterized using the interpolated objects as:

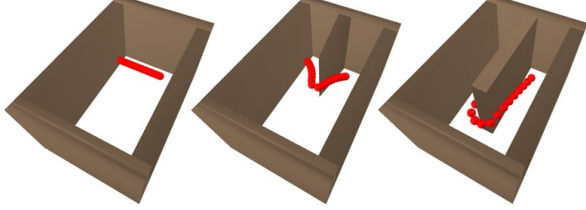
$$\begin{aligned} \mathcal{O} \left(\text{SDF}_{\mathcal{V}_k + \mathcal{V}_k^l}^\alpha \right) &= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{j=1}^{n_l} v_{v_{\sigma(j)} \rightarrow v_j^l}^\alpha \\ &= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{j=1}^{n_l} \left\{ v_{v_{\sigma(j)} \rightarrow v_j^l}^\alpha \cap (v_{\sigma(j)} \cup v_j^l) \right\} \end{aligned} \quad (7)$$

$$= \mathcal{O}_{\mathcal{V}_k} \cup \bigcup_{j=1}^{n_l} \left\{ v_{v_{\sigma(j)} \rightarrow v_j^l}^\alpha \cap v_j^l \right\} \quad (8)$$

The transformation to (7) is derived from the property (4), while (8) is derived from the fact that $v_{\sigma(j)} \in \mathcal{V}_k$. Let us denote $v_j^\alpha := v_j^l \cap v_{v_{\sigma(j)} \rightarrow v_j^l}^\alpha$ for simplicity. Then, the



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



(b) Using our framework, the path can be maintained in free space with continuous update, leading to successful path planning.

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

occupied space (8) can be interpreted as a gluing of a set $\{v_1^\alpha, \dots, v_{n_l}^\alpha\}$. The set $\{v_1^\alpha, \dots, v_{n_l}^\alpha\}$ is a leaf set, since each v_j^α does not intersect with one another, and intersects with only $v_{\sigma(j)}$ among \mathcal{V}_k :

$$\begin{aligned} v_i^\alpha \cap v_j^\alpha &\subseteq v_i^l \cap v_j^l = \emptyset \\ v_j^\alpha \cap v_{\sigma(j)} &\supseteq v_j^l \cap v_{\sigma(j)} \neq \emptyset \\ v_j^\alpha \cap v_{\sigma(i)} &\subseteq v_j^l \cap v_{\sigma(j)} = \emptyset \end{aligned}$$

for all $i \neq j$. Since gluing of a leaf set do not create or delete cycles, thus the occupied space (8) preserves homotopy with \mathcal{V}_k for all $\alpha \in [0, 1]$. Consequently, the homotopy of the occupied space for the interpolated collision constraint (6) is preserved throughout the interpolation.

D. Generation of Leaf Set Sequences

The initial objects and the sequence of the additions can be determined automatically using Algorithm 1. Starting with the total set of objects, the algorithm identifies a leaf set \mathcal{V}_k^l at each iteration, where k represents the current step of sequence. The leaf set is formed by adding objects which do not intersect with each other but uniquely intersect with \mathcal{V} (Line 7-11). This subset \mathcal{V}_k^l is then removed from \mathcal{V} (Line 12), and the process repeats with the next value of k (Line 13) until no further leaf sets can be identified. The final sequence of leaf sets is established by reversing the order of \mathcal{V}_k^l (Line 15) about k . The remaining objects \mathcal{V} , after all leaf sets have been removed, become the initial objects \mathcal{V}_{init} (Line 16).

This algorithm is designed to leave as few objects as possible as initial objects. However, if many objects need

Algorithm 1 Sequences of leaf sets

```

1: Input: Decomposed objects  $\mathcal{V}_{tot}$ 
2: Output: Sequence of leaf sets  $\mathcal{V}^l$ , Initial objects  $\mathcal{V}_{init}$ 
3:  $\mathcal{V} \leftarrow \mathcal{V}_{tot}$ 
4:  $k \leftarrow 1$ 
5: repeat
6:   Initialize  $\mathcal{V}_k^l = []$ 
7:   for  $v \in \mathcal{V}$  do
8:     if  $\mathcal{V}_k^l \cup v$  satisfies Condition 1 then
9:       Append  $v$  to  $\mathcal{V}_k^l$ 
10:    end if
11:  end for
12:   $\mathcal{V} \leftarrow \mathcal{V} \setminus \mathcal{V}_k^l$ 
13:   $k \leftarrow k + 1$ 
14: until  $|\mathcal{V}_k^l| = 0$  or  $|\mathcal{V}| = 0$ 
15:  $\mathcal{V}^l = \{\mathcal{V}_k^l, \mathcal{V}_{k-1}^l, \dots, \mathcal{V}_1^l\}$ 
16:  $\mathcal{V}_{init} = \mathcal{V}$ 

```

Algorithm 2 Narrow passage path planning

```

1: Input: Total set of nodes  $\mathcal{V}_{tot}$ 
2: Determine  $\mathcal{V}^l, \mathcal{V}_{init}$  from Alg. 1
3: Initialize environment  $\mathcal{V}_1 = \mathcal{V}_{init}$  and path  $X_{1:T}$ 
4: for  $k = 1, \dots, |\mathcal{V}^l|$  do
5:    $\alpha = 0$ 
6:   while  $\alpha < 1$  do
7:     Refine the path  $X_{1:T}$  by optimization (10)
8:      $\alpha \leftarrow \min(\alpha + \Delta\alpha, 1)$ 
9:   end while
10:   $\mathcal{V}_{k+1} = \mathcal{V}_k \cup \mathcal{V}_k^l$ 
11: end for
12: return  $\mathcal{V}_{seq}$ 

```

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 leaf set in Lines 7-11, which will not affect the subsequent path planning framework.

III. PATH PLANNING USING COLLISION CONSTRAINT INTERPOLATION

A typical path planning optimization can be formulated as:

$$\begin{aligned} \min_{X_{1:T}} \quad & \sum_{t=1}^{T-1} \|X_{t+1} - X_t\|^2 \\ \text{s.t.} \quad & \min_{x \in W(X_t)} \text{SDF}_{\mathcal{V}}(x) \geq \hat{d}, \quad t = 1, \dots, T \end{aligned} \quad (9)$$

where $X_{1:T}$ is the optimization variable representing configuration at the t -th timestep, T is the total number of waypoints, $W(X_t)$ is the surface of the robot at each waypoint X_t , and $\hat{d} \in \mathbb{R}^+$ is the predefined safe distance. Here, we aim to generate a series of subproblems tailored to narrow passage problem using the collision constraint interpolation in Sec. II.

The overall path planning algorithm is explained in Algorithm 2. We first decompose the environment into a set

of convex objects, and initialize the path by ignoring the constraint of (9). Then, the leaf set addition sequence and initial objects can be identified using Algorithm 1. Then, the subproblems are defined by substituting the collision avoidance constraint in (9) with the interpolated collision constraint:

$$\begin{aligned} & \min_{X_{1:T}} \sum_{t=1}^{T-1} \|X_{t+1} - X_t\|^2 \\ \text{s.t. } & \min_{x \in W(X_t)} \text{SDF}_{\mathcal{V}_k + \mathcal{V}_k^t}^\alpha(x) \geq \hat{d} \end{aligned} \quad (10)$$

For each k -th sequence, α is gradually increased from 0 in increments of $\Delta\alpha$, creating a series of subproblems. In each subproblem, the collision constraint becomes progressively stricter, guiding the optimization toward a solution to the narrow passage planning problem (9).

To solve the subproblem (10), we employ Sequential Quadratic Programming (SQP) by linearizing the problem. The resulting Quadratic Programming (QP) at each iteration is solved using SubADMM [16], which is particularly adept at stably and efficiently solving conflicting constraints common in narrow passage path planning.

We need to detect the collision to solve the problem (10). Even though the environment changes with α , by leveraging the SDF interpolation, we can efficiently manage the collision detection. By separating the interpolated collision constraint (6) into two parts, we can express the constraint as follows:

$$\text{SDF}_{\mathcal{V}}(x) \geq \hat{d} \quad (11)$$

$$\text{SDF}_{v_{\sigma(j)} \rightarrow v_j^t}^\alpha(x) \geq \hat{d}, \quad j \in \{1, \dots, m\} \quad (12)$$

where (11) represents the collision constraint for the current environment \mathcal{V} and (12) corresponds to the collision constraint for the interpolated convex objects.

The collision detection for the interpolated convex objects (12) can be performed using the Frank-Wolfe algorithm [17]. That is, from the given triangular mesh of the robot $W(X_t) = \{W_1, \dots, W_{n_w}\}$ at waypoint X_t , collision detection can be conducted as:

$$x^* = \arg \min_{x \in W_i} \text{SDF}_{v_{\sigma(j)} \rightarrow v_j^t}^\alpha(x)$$

where W_i is the i -th triangular mesh. Due to the convexity of the interpolated SDF (2), the solution converges to the global optimum at a sublinear rate. One key advantage of employing SDF-based interpolation lies in its ability to handle collision detection without requiring the computation of the geometry of the interpolated environment.

Unlike the interpolated convex objects, where the geometry changes with α , the objects in \mathcal{V} have fixed geometries that can be represented with meshes. This allows for faster collision detection methods beyond Frank-Wolfe such as the Gilbert-Johnson-Keerthi with Expanding Polytope Algorithm (GJK-EPA) [18].

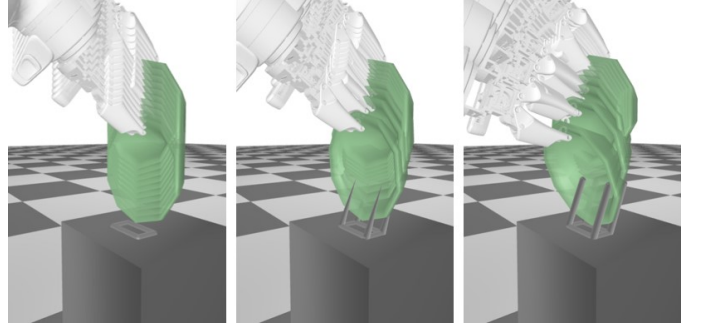


Fig. 6: Results of a successful path planning using the proposed framework for the dish insertion task.

IV. EVALUATIONS

We test our framework in three different scenes, 1) placing dishes on a drying rack, 2) extracting a box from a narrow gap, and 3) maze navigation of nonholonomic system. The first two scenarios are compared against the results obtained using baseline planners from TrajOpt [10], OMPL [19] and CHOMP [8]. The timeout for the OMPL was set to 30 seconds and 50 seconds for each respective scenario.

A. Placing Dishes on the Rack

Inserting a dish into a narrow gap of a drying rack is challenging for a manipulator [20]. The thinness of the rack makes deep penetration more likely, resulting in being stuck in infeasibility. For the same reason, even achieving the feasible goal position is challenging for this scenario.

We define the objective function as the distance to an approximate reference pose located at the center of the rack, facing horizontal direction. Additionally, the Cartesian path length objective function and a hard constraint on the initial joint position are incorporated. 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.

Table I compares the result of the optimization with and without collision constraint interpolation. Success time shows the mean and standard deviation of the elapsed time of success cases, while total time shows both success and failure cases. The tests are conducted using combinations of three different shapes of dishes and 20 different racks. The results indicate that our method outperforms the one without collision constraint interpolation in both success rate and computation time.

Method	Success	Total time (s)	Success time (s)
Proposed	58/60	3.56 ± 0.69	3.95 ± 0.68
Without interpolation	23/60	7.06 ± 0.33	7.42 ± 0.27

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

Conventional planning methods typically require a predefined goal pose. To compare our proposed framework with

Method	Success	Total time (s)	Success time (s)
Proposed	20/20	3.28 ± 0.61	3.28 ± 0.61
RRTConnect	9/20	20.08 ± 11.18	10.45 ± 10.32
BiTRRT	10/20	22.21 ± 9.91	16.24 ± 11.10
TRRT	2/20	26.37 ± 9.02	1.36 ± 0.00
BiEST	4/20	23.42 ± 9.91	16.24 ± 11.10
BMFT	2/20	21.34 ± 7.86	10.85 ± 8.33
PRMstar	0/20	31.67 ± 0.90	-
LazyPRM	0/20	30.03 ± 0.00	-
KPIECE	1/20	22.20 ± 9.09	8.48 ± 0
BKPIECE	4/20	21.34 ± 7.86	10.85 ± 8.33
TrajOpt	0/20	4.76 ± 1.27	-
CHOMP	2/20	22.74 ± 2.58	15.28 ± 0.03

TABLE II: Comparison with other methods for the task of placing dish, showing the planning time and success rate, with the goal position provided by our framework.

Method	Success	Total time (s)	Success time (s)
Proposed	30/30	6.52 ± 0.69	6.52 ± 0.69
RRTConnect	1/30	48.83 ± 3.54	32.91 ± 0.00
BiTRRT	19/30	30.33 ± 14.23	23.15 ± 12.80
TRRT	0/30	50.03 ± 0.00	-
BiEST	0/30	50.07 ± 0.16	-
BMFT	1/30	49.51 ± 4.98	32.41 ± 0.00
PRMstar	0/30	50.28 ± 0.41	-
LazyPRM	0/30	50.04 ± 0.01	-
KPIECE	0/30	50.04 ± 0.015	-
BKPIECE	0/30	50.07 ± 0.14	-
TrajOpt	17/30	3.41 ± 1.09	2.71 ± 0.93
CHOMP	0/30	32.66 ± 11.20	-

TABLE III: Comparison with other methods for the task of extracting tool from a narrow gap, showing the planning time and success rate.

existing methods, we used the feasible final pose obtained by our framework as the goal position for the other methods. We tested twelve different shapes of dish racks, with the results presented in Table II. As also shown in [2], conventional sampling-based methods—except for BiTRRT [21]—faced significant challenges in solving the narrow passage problem. 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. Note that in the case of TrajOpt, continuous collision detection is performed by assuming a convex hull between waypoints. However, this results in an overly conservative over-approximation for the non-convex geometry of the dish, leading to failure in successful execution.

B. Extraction of Tool from a Narrow Gap

Taking tool out through a narrow gap is a challenge task for a manipulator, especially when the size of the tool is large, or the obstacles are thin. Our objective is to optimize the manipulator path, starting from a pose that grasping the object, and extracting it out through a narrow gap. Fig 1 shows the result of the optimization.

We introduced slight randomness into the environment configuration to create 30 environments. Each planner was

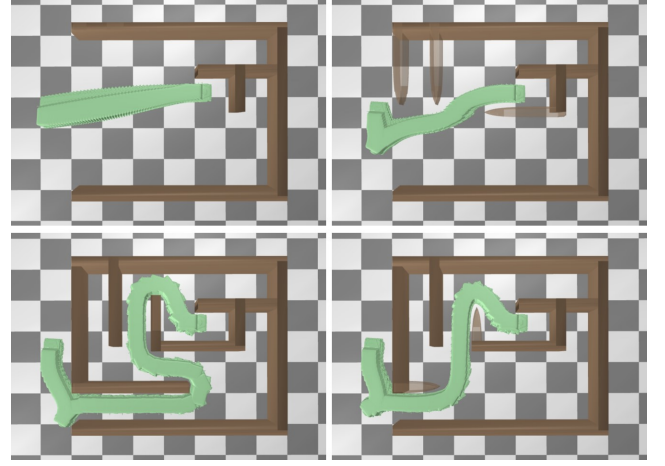


Fig. 7: Process of the path planning optimization of a nonholonomic system in a maze (clock-wise).

tested under these conditions. As shown in Table III, CHOMP and sampling methods had low success rates and longer planning times compared to our approach. While TrajOpt achieved higher success rates and shorter planning times than the sampling methods, its success rates were still lower than our framework, which successfully planned in all configurations.

C. Nonholonomic Navigation of Maze

Nonholonomic navigation of maze has been studied using various approaches such as RRT-based nonholonomic planning [22]. However, once the volume of the vehicle is considered, the corridors in the configuration space become extremely narrow, causing a significant loss of scalability of sampling-based approaches. Our methodology can augment nonholonomic constraints within a general optimization framework and address the narrow corridor problem through the collision constraint interpolation. The result of the optimization is illustrated in Fig. 7. Starting from an environment with enlarged free space, the path of the vehicle is effectively optimize to navigate through the maze.

V. CONCLUSIONS

We present a path planning framework specifically designed to address the challenges posed by narrow passage, but it has several limitations. Firstly, objects can only be added when contained in a leaf set, which may limit the applicability of the framework in environments with complex topologies. Moreover, the collision constraint interpolation incurs additional overhead due to the convex decomposition of the environment. Additionally, as the problem is solved iteratively by increasing the interpolation variable α , this method can be time-consuming in scenarios where extremely narrow passages are not present. Finally, the optimization-based nature of our methodology risks getting stuck in local minima. Future work includes analyzing homotopy and extending the framework to more general and complex environments.

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] N. Hiraoka, H. Ishida, T. Hiraoka, K. Kojima, K. Okada, and M. Inaba. Sampling-based global path planning using convex polytope approximation for narrow collision-free space of humanoid. *International Journal of Humanoid Robotics*, page 2450005, 2024.
- [5] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] K. Erleben. Methodology for assessing mesh-based contact point methods. *ACM Transactions on Graphics*, 37(3):1–30, 2018.
- [13] 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.
- [14] 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.
- [15] G. Turk and J. F. O’Brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH Courses*, pages 13–es. 2005.
- [16] 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.
- [17] 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.
- [18] G. Van Den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*, volume 170, 2001.
- [19] D. Coleman, I. Sucas, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.
- [20] J. Lee, M. Lee, and D. Lee. Uncertain pose estimation during contact tasks using differentiable contact features. *Robotics: Science and Systems*, 2023.
- [21] D. Devaurs, Thierry Siméon, and Juan Cortés. Enhancing the transition-based rrt to deal with complex cost spaces. In *IEEE international conference on robotics and automation*, pages 4120–4125, 2013.
- [22] 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.