2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Value Iteration Algorithm for Solving Shortest Path Problems with Homology Class Constraints

Wenbo He, Yunshen Huang, Jinran Qie, and Shen Zeng

***Abstract*— Path planning is a fundamental problem in robotics that aims to find an optimal path for a system to move on while avoiding obstacles in the environment. Often, a feasible path connecting the start and target point with the shortest length is highly desirable. Additionally, in scenarios such as drone racing or surveillance, topology constraints may arise. In this paper, we propose a novel method to address the shortest path problem with homology class constraints in both 2D and 3D environments. We first define the phase change of the path with respect to 2D obstacles and then apply the same technique to a class of super-toroid obstacles compressed by an embedding map. To synthesize the shortest path, we leverage the visibility graph and the Value Iteration Algorithm (VIA). Finally, we demonstrate the effectiveness of our approach with various simulation examples.**

I. Introduction

Path planning problems consider synthesizing a collision-free path within which to navigate a moving object in a cluttered environment. This type of problem is crucial for controlling mobile robots and autonomous systems and has been intensively studied [1]. Given a system with very simple dynamics, various sample-based methods can be deployed for handling nonconvex environments, such as probabilistic roadmaps (PRM) [2] and rapidly-exploring random trees (RRT) [3]. Some variations further take optimality into account, such as RRT* [4]. When system dynamics become more complicated, some optimization-based methods, which include iterative (LQR) [5] and sequential homotopy quadratic programming (SHQP) [6], can be leveraged. Despite the successful applications, they are unable to handle the cases where topological constraints are imposed, such as homotopy class constraints and homology class constraints.

Two trajectories belong to the same homotopy class if they share the same initial and target state, and they can continuously deform to each other without intersecting any obstacles. Homotopy class constraints can be encountered in, for instance, drone racing, where agents are required to fly through gates in a particular order. Multiple works are designed for solving such planning problems with homotopy class constraints. For instance, search-based methods [7] [8] are able to classify homotopy classes and synthesize feasible trajectories fulfilling the topological requirements. For non-trivial system dynamics, effective approaches like Mixed-Integer Quadratic Program (MIQP)-based method [8], Auxiliary Energy Reduction (AER) [9], and Homotopy Method for Homotopy Class Constraints (HMHCC) [10] are proposed.

Department of Electrical and System Engineering, Washington University, St. Louis, MO 63130, USA, {wenbo.he, yunshen.huang, jinran.qie, s.zeng}@wustl.edu. This work was supported by the NSF grant CMMI-1933976.

Generally speaking, two trajectories belong to the same homology class if they are homotopy equivalent regarding every single obstacle. Again, in drone racing, trajectories that connect the same initial and target locations and pass through every gate in arbitrary order constitute a homology class. Although homology class constraints can be considered as the loose representation of homotopy class constraints, the corresponding optimization is much harder to address, as the feasible solution set is no longer compact. Therefore, the global search-based algorithm is preferable. Pioneering methods based on H-signature are proposed to generate optimal trajectories under homology class constraints for both 2D [11] and 3D [12] environments. Despite the capability of dealing with obstacles with general shapes, H-signature-based methods have to be defined differently depending on the environmental dimensionality, which hinders efficiency. In this paper, we proposed a novel approach that solves the constrained shortest path problem for both 2D and 3D environments in a unified manner. For 3D cases, we first compress the super-toroid-shaped obstacles into a 2D counterpart. We then classify the homology class of the 2D trajectory by adopting the idea of phase change. Having done so, we are able to compute the shortest path fulfilling the topological requirement using a visibility graph and VIA. The analysis of our method suggests the potential generalization to higher dimensional environments with obstacles that are properly described.

This paper is organized as follows. Section II defines the homotopy and homology class, as well as obstacles in different dimensions. Then we introduce the proposed method in both 2D and 3D environments and the Value Iteration Algorithm in Section III. Numerical Results are shown in Section IV and we conclude the paper in Section V.

II. Mathematical Preliminaries and Problem Statement

A. Homotopy and Homology Classes for Static Obstacles

In this section, we begin with the definition of homotopy class for trajectories and depict the connection between homotopy and homology. Two continuous trajectories belong to the same homotopy class if and only if they have identical start and target points and they can be smoothly deformed into one another without intersecting any obstacles, as shown in Figure 1. The formal definition is shown below.

Definition 1. *Two continuous trajectories* $r_1(t) : [0,1] \to \mathbb{R}^n$ *and* $r_2(t) : [0,1] \to \mathbb{R}^n$ *belong to the same homotopy class if and only if* $r_1(0) = r_2(0)$, $r_1(1) = r_2(1)$, *and there*

exists a continuous function $H(s,t) : [0,1] \times [0,1] \to \mathbb{R}^n$ *which satisfies* $H(0,t) = r_1(t)$, $H(1,t) = r_2(t)$ *and* $H(s,t)$ *never intersects any obstacles for all* $s \in [0,1]$ *and* $t \in [0,1]$.

Considering a 2-dimensional space with a single obstacle, the homotopy class of a trajectory can be quantified from the perspective of phase change. Given a trajectory $r(t) : [0,1] \to \mathbb{R}^2$ and a fixed point o_i representing the center point of the obstacle, the responding phase change regarding the obstacle is defined as

$$p(r) = \theta(r(1) - o_i) - \theta(r(0) - o_i), \tag{1}$$

where $\theta : \mathbb{R}^2 \to \mathbb{R}$ is the unwrapped angle, which is the unwrapped version of $\arctan : \mathbb{R}^2 \to [0, 2\pi)$ that ensure $\theta(p(\cdot))$ is continuous if $p(\cdot)$ is continuous. It implies that two trajectories with identical start and target points belong to the same homology class if they own the same phase change.

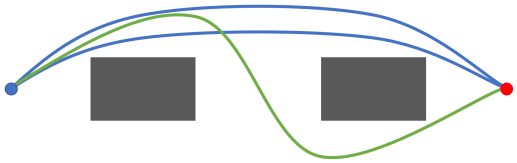

Fig. 1. The three curves are the trajectories connecting identical starting and target points, and grey rectangles are obstacles. According to the definition of the homotopy class, the two blue trajectories are homotopy equivalent, but the green one belongs to a different homotopy class.

In a cluttered environment, the phase change with respect to n obstacles is given by $p(r) = [p_1(r), \ldots, p_n(r)]^T$, where $p_i(r)$ is the phase change regarding the ith obstacle. Generally, for an arbitrary dimensional space with multiple obstacles, trajectories belong to the same homology class if they have the same $p(r)$, i.e., they belong to the same homotopy class regarding every individual obstacle. However, having the same phase change is only a necessary condition for homolopy as shown in Figure 2, and the formal definition is given below.

Definition 2. *Two continuous trajectories* $r_1(t) : [0,1] \to \mathbb{R}^n$ *and* $r_2(t) : [0,1] \to \mathbb{R}^n$ *belong to the same homology class if and only if* $r_1(0) = r_2(0)$, $r_1(1) = r_2(1)$, *and* $r_1(t)$ *together with* $r_2(t)$ *with the opposite orientation forms the complete boundary of a 2-dimensional manifold embedded in* $\mathbb{R}^n$ *not containing any obstacles.*

Although homology is a loose representation of homotopy in many applications, finding the optimal trajectory belonging to a specified homology class is even more challenging than that of the homotopy counterpart. It can be seen from the fact that a homotopy class can be treated as a compact set, as every element can continuously deform to one another. Therefore, starting from a random feasible trajectory, one can reach the local or even global optimum of such a homotopy class by leveraging the gradient information. Nevertheless, a homology class is usually a disconnected set that contains various homotopy classes, which prevents the implementation of the gradient method, and thus a global searching mechanism is needed. The objective of this paper is to design the shortest trajectory that is obstacle-free and belongs to a particular homology class in both 2D and 3D space.

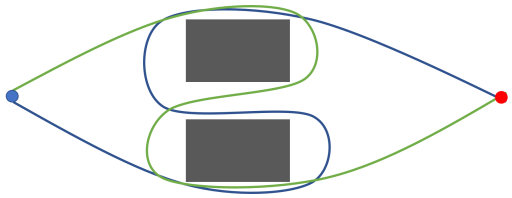

Fig. 2. The two curves are the trajectories connecting identical starting and target points. They belong to the same homology class but different homotopy classes.

B. Static Obstacle Representations

For practical purposes, we consider M numbers of static obstacles in i) a 2-dimensional (2D) and ii) a 3-dimensional (3D) cluttered environment. In case i), though the proposed method can handle obstacles with arbitrary shapes, we approximate them using super-ellipse [6] that is aligned with the case of the 3D environment. In case ii), we consider a type of obstacle with a hole connecting the two opposite facets, which emulates the obstacle that a moving object needs to pass through.

In 2D environments, the collision avoidance constraints regarding the ith obstacle can be written as

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} := M_i \, \mathrm{diag}(e_{ix}, e_{iy}) \begin{bmatrix} x \\ y \end{bmatrix} + b_i, \tag{2}$$

$$\hat{x}^{k_i} + \hat{y}^{k_i} - {R_i}^{k_i} \geq 0, \tag{3}$$

where $[x, y]$ denotes the allowed spatial trajectory of the dynamical system, M_i is the rotation matrix, $e_{ix}, e_{iy} \geq 1$ represent the spatial expansion factor along each axis, b_i is the offset of the center of the obstacle, $R_i > 0$ is a size constant and the exponent k_i is a positive even number, which regulates the shape of the obstacle. Particularly, when $k_i = 2$, the obstacle is a circle or ellipse, otherwise, it would be a rounded square.

In 3D environments, we consider a group of obstacles with super-toroid shapes, where the ith obstacle is described by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} := M_i \, \mathrm{diag}(e_{ix}, e_{iy}, e_{iz}) \begin{bmatrix} x \\ y \\ z \end{bmatrix} + b_i, \tag{4}$$

$$\hat{z}^{m_i} + (R_i - (\hat{x}^{n_i} + \hat{y}^{n_i})^{\frac{1}{n_i}})^{m_i} - {r_i}^{m_i} \geq 0, \tag{5}$$

where R_i is the distance from the center of the tube to the center of the obstacle, r_i is the radius of the tube, and the rest of the parameters are defined in a similar way to that in (2). It is noted that m_i and n_i have to be positive even numbers. Some examples of the obstacle are shown in Figure 3

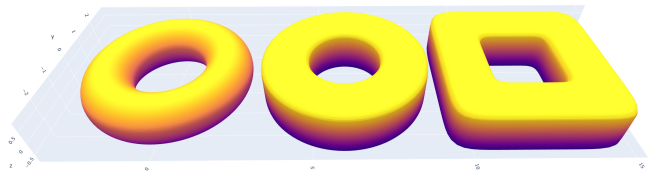

Fig. 3. 3-dimensional obstacles with $(n, m) = (2, 2), (2, 8), (8, 8)$ from left to right, respectively.

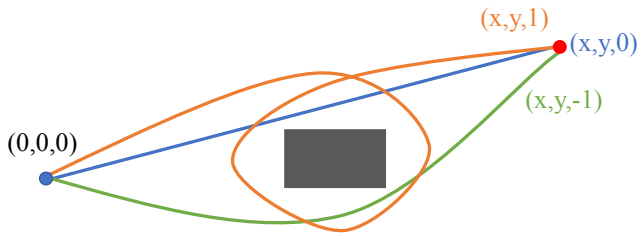

Fig. 4. Three different paths connecting $(0, 0)$ and (x, y) belong to different homology classes, where the corresponding nodes at (x, y) are marked as well.

III. Methods

In this section, we describe our strategy to synthesize the shortest obstacle-free path within a specific homology class. Our method relies on the phase change representation of the homology class that is well-defined in 2D space, which will be illustrated in Section III-A. In Section III-B, we prove that the interested type of 3D obstacles can be simply compressed into a 2D space. In both cases, we transform feasible paths into a graph and find the shortest one using the value iteration shown in Section III-C.

A. 2D Environment

To better present the corresponding homotopy class of a path, we associate a phase change with each point of the path. More specifically, given a path and M obstacles, we define a node of the path as a vector $(x, y, p_1, \ldots, p_M)$, where (x, y) represents the position of the node and $p_i \in \mathbb{R}$ denotes the corresponding phase change regarding the ith obstacle. Because of the conservation of the phase change, given a specified path, the phase change regarding an obstacle only relies on the number of the encirclement around it that must be an integer. Therefore, the difference in the phase change of the two paths with the same ending points has to be an integral multiple of 2π. As a result, we are able to define a straight line, which connects the given start point $\mathbf{x}_{\text{start}}$ and (x, y), as a base path with a base phase change defined as $(\bar{p}_1, \ldots, \bar{p}_M)$. It is noted that such a base path does not have to be feasible in terms of obstacles. According to the base path, the node located at (x, y) of any feasible path, which starts from $\mathbf{x}_{\text{start}}$, can can be expressed as $(x, y, \bar{p}_1 + s_1 2\pi, \ldots, \bar{p}_M + s_M 2\pi),\ s_i \in \mathbb{Z}$. For simplicity, we use $(x, y, s_1, \ldots, s_M)$ to define the state in the rest of the paper and $(s_1, \ldots, s_M)$ is the homology class label, where an example is shown in Figure 4. Likewise, if the line connects $(x^a, y^a, s_1^a, \ldots, s_M^a)$ and $(x^b, y^b, s_1^b, \ldots, s_M^b)$, we define the label-wise phase change as $(\Delta s_1, \ldots, \Delta s_M) = (s_1^a - s_1^b, \ldots, s_M^a - s_M^b)$.

Equipped with the definition above, we proposed a graph search-based method to find the shortest path for moving an agent from the start point to the target, where the agent is assumed to be holonomic, i.e., any path is dynamically feasible to the agent. Specifically, we consider a visibility graph that is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of sampled states and $\mathcal{E}$ contains obstacle-free connections between these states. Given an interested point (x, y), we obtain $(x, y, s_1, \ldots, s_M)$ as sampled states with $s_i \in \{-\Gamma, -\Gamma + 1, \ldots, \Gamma\}$ and $\Gamma \in \mathbb{Z}^+$ is the hyper-parameter that determines the size of the sample space. Likewise, we say a state is legal iff $-\Gamma \leq s_i \leq \Gamma$ for all $i \in \{1, \ldots, M\}$. As such, given N interested points, the cardinality of the sample space is $|\mathcal{V}| = N(2\Gamma + 1)^M$, whose value significantly impacts the time complexity of graph search-based algorithms. To reduce the number of interested points without highly degenerating the performance of the proposed method, we sample locations around obstacles rather than discretize the space into a grid. Around the obstacles described in Section II-B, we adopt the sampling strategy depicted in the following way

$$\begin{bmatrix} x \\ y \end{bmatrix}_{\text{sample}} = \text{diag}(1/e_{ix}, 1/e_{iy}) M_i^{-1} \left(\frac{R_i + \delta}{(\cos(\theta)^{k_i} + \sin(\theta)^{k_1})^{1/k_i}} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} - b_i \right), \tag{6}$$

where $\theta \in \{0, 2\pi/N_i, \ldots, 2(N_i - 1)\pi/N_i\}$ is the sampling angle, N_i is the number of samples around the obstacle, and $\delta > 0$ is a hyper-parameter that ensures the sampled points residing outside the obstacle. The algorithm for building the visible graph is summarized in Algorithm 1. Note that the graph assumes that each state is self-connected. Given the graph $\mathcal{G}$, the shortest path can be obtained by value iteration that will be described in Section III-C.

Algorithm 1 Construct Visibility Graph

Require: Obstacle description and hyper-parameter Γ
Initialize an empty graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a potions set $\mathcal{L}$
Sampled points according to (6), and store them in $\mathcal{L}$
Push the start and target points into $\mathcal{L}$
for each pair of points $((x^a, y^a), (x^b, y^b))$ from $\mathcal{L}$
    if the straight line connecting the pair is infeasible
        continue
    Get phase change $(\Delta s_1, \ldots, \Delta s_M)$ of the straight line
    for each legal $(s_1, \ldots, s_M)$
        if $(s_1 + \Delta s_1, \ldots, s_M + \Delta s_M)$ is also legal
            Add the edge between $(x^a, y^a, s_1, \ldots, s_M)$ and $(x^b, y^b, s_1 + \Delta s_1, \ldots, s_M + \Delta s_M)$ to $\mathcal{G}$
return $\mathcal{G}$

B. 3D Environment

It is noted that the graph-based method proposed above depends on the phase change for classifying homology classes.

Hereby, it is severely restricted and thus only applicable to planar systems. To generalize the method to 3-dimensional, as defined in Section II-B, we consider a certain type of 3D obstacle equipped with a hole, where the agent can pass through. Because the coordinate transformation between $(\hat{x}, \hat{y}, \hat{y})^\top$ and $(x, y, z)^\top$ is invertible in (4), in this section and without loss of generality, we only discuss normalized obstacles

$$z^{m_i} + (R_i - (x^{n_i} + y^{n_i})^{\frac{1}{n_i}})^{m_i} - {r_i}^{m_i} \geq 0. \tag{7}$$

We consider an embedding mapping $f_i : \mathbb{R}^3 \to \mathbb{R}^2$

$$f_i\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} (x^{n_i} + y^{n_i})^{\frac{1}{n_i}} \\ z \end{bmatrix} := \begin{bmatrix} x_i^e \\ y_i^e \end{bmatrix}, \tag{8}$$

which compresses the ith 3D obstacle to a super-ellipse centered at $(R_i, 0)$ with the radius r_i and the exponent m_i. As a result, the phase change technique proposed in Section III-A can be safely applied.

Lemma 1. *Given a trajectory $r(t) : [0, 1] \to \mathbb{R}^3$ and the super-toroid obstacle, the trajectory is obstacle-free iff the embedded trajectory $(x_i^e(t), y_i^e(t))^\top = f_i(r(t))$ is obstacle-free for all $t \in [0, 1]$ in the embedding space:*

$$(R_i - x_i^e(t))^{m_i} + y_i^e(t)^{m_i} - r_i^{m_i} \geq 0.$$

Proof. Replacing $x_i^e(t)$ and $y_i^e(t)$ by $(x(t)^{n_i} + y(t)^{n_i})^{\frac{1}{n_i}}$ and $z(t)$ in (2), respectively, the obstacle-free condition for 2D super-ellipse is identical to the one of that in 3D space. □

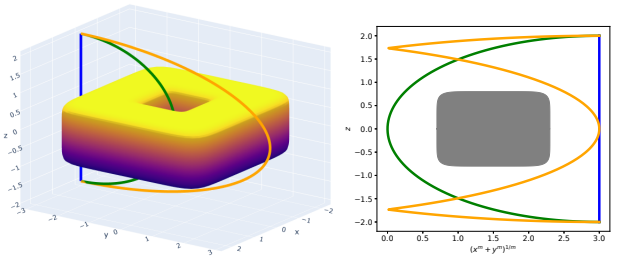

Fig. 5. A super-toroid obstacle and three trajectories are shown in the left figure and their shapes in the embedding space are shown in the right figure, where the homotopy property of three trajectories still holds.

As such, one can easily evaluate the phase change of a trajectory and further the homotopy relationship of multiple trajectories in the embedding space, which is shown in Figure 5. Having the same phase change in the embedding space is the necessary and sufficient condition for two trajectories to be within the same homotopy class when only one obstacle exists as shown in the corollaries below. If trajectories are homotopy equivalent towards every single obstacle, then they belong to the same homology class.

Corollary 1.1. *If two continuous trajectories $r_1(t) : [0, 1] \to \mathbb{R}^3$ and $r_2(t) : [0, 1] \to \mathbb{R}^3$ belong to the same homotopy class regarding the super-toroid obstacle, then their embedding trajectories $r_1^e(t)$ and $r_2^e(t)$ are also homotopy equivalent regarding the embedded obstacle.*

Proof. There exist a continuous function $H(s, t) : [0, 1] \times [0, 1] \to \mathbb{R}^3$ that $H(s, t)$ never inside the obstacle, $H(0, t) = r_1(t)$ and $H(1, t) = r_2(t)$ because r_1 and r_2 are homotopy equivalent. We define $H^e(s, t) = f_i(H(s, t))$, then $H^e(0, t) = r_1^e(t)$ and $H^e(1, t) = r_2^e(t)$. According to Lemma 1, $H^e(s, t)$ is obstacle-free, and the function $H^e(s, t)$ is still continuous because the embedding function f_i is continuous. Therefore $r_1^e(t)$ and $r_2^e(t)$ are also homotopy equivalent. □

Corollary 1.2. *If two continuous trajectories $r_1(t) : [0, 1] \to \mathbb{R}^3$ and $r_2(t) : [0, 1] \to \mathbb{R}^3$ have the same initial and terminal points, and their embedded trajectories $r_1^e(t)$ and $r_2^e(t)$ are homotopy equivalent regarding the compressed obstacle, then $r_1(t)$ and $r_2(t)$ belong to the same homotopy class towards the original 3-dimensional super-toroid obstacle.*

Proof. We define the complementary embedding mapping

$$f_i^c\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} (x^{n_i} + y^{n_i})^{\frac{1}{n_i}} \\ z \\ \arctan(y, x) \end{bmatrix} := \begin{bmatrix} x_i^e \\ y_i^e \\ z_i^c \end{bmatrix},$$

then f_i^c is the generalized cylindrical coordinate transformation. We kindly assume $r_1(t)$ and $r_2(t)$ never attach the z-axis, then their complementary trajectories $r_1^c(t) = f_i^c(r_1(t))$ and $r_2^c(t) = f_i^c(r_2(t))$ are also continuous and well-defined. Because $r_1^e(t)$ and $r_2^e(t)$ are homotopy equivalent, there exists a continuous function $H(s, t)$ that is obstacle-free and connects $r_1^e(t)$ and $r_2^e(t)$. Then we define

$$H^c(s, t) = \bar{f}_i^c\left(\begin{bmatrix} H(s, t)_1 \\ H(s, t)_2 \\ (1 - s) \cdot z_i^c(r_1(t)) + s \cdot z_i^c(r_2(t)) \end{bmatrix}\right),$$

where $\bar{f}_i^c : \mathbb{R}^3 \to \mathbb{R}^3$ is the mapping from the generalized cylindrical coordinate to the original Cartesian coordinate, thus $H^c(0, t) = r_1(t)$ and $H^c(1, t) = r_2(t)$. According to Lemma 1, $H^c(s, t)$ is obstacle-free. Because $\bar{f}_i^c$ is continuous, $H^c(s, t)$ is also continuous. Hence $r_1(t)$ and $r_2(t)$ are homotopy equivalent. □

Similar to the 2D case, the state in a 3D environment is defined as $(x, y, z, s_1, \ldots, s_M)$, where (x, y, z) is the space location and $(s_1, \ldots, s_M)$ is the homology class label calculated in the 2-dimensional embedding space. To decrease the number of sampled states, we uniformly sample nodes around the obstacle as shown in Algorithm 2, and then construct the graph in the same way as Algorithm 1 except that the trajectory and obstacles need to be transformed to the embedding space according to (8).

C. Optimization Method: Value Iteration

Given an undirected graph built in the previous sections, the Value Iteration Algorithm (VIA) will then be implemented to yield the shortest path. Although VIA has been widely applied in stochastic shortest path problems, it is less efficient compared with the Dijkstra algorithm in deterministic situations. But we will show that the format of VIA

can be embarrassingly paralleled and therefore outperform the Dijkstra algorithm on multi-core CPU or GPU.

Algorithm 2 Sample locations around the 3D obstacle

Require: obstacle parameters, hyper-parameter δ
1: Initialize empty set S
2: **for** $\theta \in \{\theta_1, \dots, \theta_n\}$ and $s \in \{s_1, \dots, s_n\}$
3: $\quad \lambda \leftarrow (r_i + \delta)/((\cos(s)^{m_i} + \sin(s)^{m_i})^{1/m_i}$
4: $\quad \mu \leftarrow ((\cos(\theta)^{n_i} + sin(\theta)^{n_i})^{1/n_i}$
5: $\quad \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \leftarrow \begin{bmatrix} (R_i + \lambda \sin(s)) \cos(\theta)/\mu \\ (R_i + \lambda \sin(s)) \sin(\theta)/\mu \\ \lambda \cos(s) \end{bmatrix}$
6: $\quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leftarrow diag(1/e_{ix}, 1/e_{iy}, 1/e_{iz}) M_i^{-1} \left(\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} - b_i \right)$
7: $\quad$ Put $(x, y, z)^T$ to S
8:**return** S

Shortest path problems based on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be converted to reinforcement learning (RL) problems, which are infinite-horizon Markov decision processes (MDP) with a terminal state. To begin with, we use the vertex set $\mathcal{G}(\mathcal{V})$ to represent the state set of RL, and $A(\mathbf{s}_i)$ to denote the feasible actions set of the state $\mathbf{s}_i$. An action $a_{i,j} \in A(\mathbf{s}_i)$ will transfer the state from $\mathbf{s}_i$ to $\mathbf{s}_j$. Moreover, the return function of the state transfer is defined as

$$R(\mathbf{s}_i, a_{i,j}) = \begin{cases} \Lambda - E(\mathbf{s}_i, \mathbf{s}_j), & \text{if } \mathbf{s}_j \text{ is } \mathbf{s}_T \\ -E(\mathbf{s}_i, \mathbf{s}_j), & \text{else} \end{cases} \tag{9}$$

where $\mathbf{s}_T$ is the terminal state, $\Lambda \in \mathbb{R}$ is a large enough value and $E(\mathbf{s}_i, \mathbf{s}_j)$ is the Euclidean distance of two states' locations. A policy $\pi(a|\mathbf{s})$ describes a deterministic desired action for each state. Starting from an initial state $\mathbf{s}_1$, the policy π provides an infinite state sequence $\{\mathbf{s}_1, \mathbf{s}_2, \dots\}$ with return value sequence $\{r_1, r_2, \dots\}$. We define $\gamma \in (0, 1)$ as the discount factor, then the value function is defined as

$$V^{\pi}(\mathbf{s}_1) = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots = r_1 + \gamma V^{\pi}(\mathbf{s}_2) \tag{10}$$

Under the optimal policy π' which maximize the value function $V^{\pi'}$ for every states, we have $V^{\pi'}(\mathbf{s}_T) = \Lambda/(1-\gamma)$ and $V^{\pi'}(\mathbf{s}_0) = \sum_{i=0}^{n} \gamma^i E(\mathbf{s}_i, \mathbf{s}_{i+1}) + \gamma^{n+1} V^{\pi'}(\mathbf{s}_T)$, where $\{\mathbf{s}_i\}_{i=0}^{n}$ forms a feasible trajectory from $\mathbf{s}_0$ to $\mathbf{s}_T$ for any $\mathbf{s}_0 \in \mathcal{V}$. If we choose γ close enough to 1, then $\{\mathbf{s}_i\}_{i=0}^{n}$ is the shortest path, and $V^{\pi'}(\mathbf{s}_0)$ approaches $V^{\pi'}(\mathbf{s}_T)$ minus the length of the shortest path from $\mathbf{s}_0$ to $\mathbf{x}_T$. Note that Λ should be larger than the longest shortest path from every state, otherwise the optimal state sequence of some states would trivially stay in place. The optimal trajectory is hereby given by

$$\mathbf{s}_{i+1} = \underset{\mathbf{s}_j}{argmax} \left(-E(\mathbf{s}_i, \mathbf{s}_j) + \gamma V^{\pi'}(\mathbf{s}_j) \right), \tag{11}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ should be connected directly in the graph and therefore $a_{i,j} \in A(\mathbf{s}_i)$.

To obtain the optimal value function $V^{\pi'}$, we start from a initial estimated value function V_0^{π} and update it iteratively:

$$V_{n+1}^{\pi}(\mathbf{s}_i) = \underset{a_{i,j} \in A(\mathbf{s}_i)}{max} \left(R(\mathbf{s}_i, a_{i,j}) + \gamma V_n^{\pi}(\mathbf{s}_j) \right), \tag{12}$$

which is referred to as Value Iteration as Algorithm 3. Although it's proven that V_n^{π} will converge to $V^{\pi'}$ for any initial estimate, the reasonable initial value will dramatically reduce the iterations it needs. In practice, we choose $V_0^{\pi}(\mathbf{s}_T) = \Lambda/(1-\gamma)$ and $V_0^{\pi}(\mathbf{s}) = 0$ for other states. Within each iteration, the updating processes of states are independent, therefore no effort is needed to separate the updating into a number of parallel tasks and reach linear speedup, which is referred to as embarrassingly parallel.

Algorithm 3 Value Iteration Algorithm (VIA)

Require: a small number θ
1: Initialize V_0^{π}, $n \leftarrow 0$
2: **do**
3: $\quad \Delta \leftarrow 0$
4: $\quad$ **for each** $\mathbf{s}_i \in \mathcal{V}$:
5: $\quad\quad$ Update $V_{n+1}^{\pi}(\mathbf{s}_i)$ according to (12)
6: $\quad\quad \Delta \leftarrow \Delta + |V_{n+1}^{\pi}(\mathbf{s}_i) - V_n^{\pi}(\mathbf{s}_i)|$
7: $\quad n \leftarrow n + 1$
8: **while** $\Delta > \theta$
3. **return** V_n^{π}

IV. Numerical Results

We evaluate the performance of the proposed method on various 2D and 3D cases, and compare the efficiency with cutting-edge algorithms [12]. For all simulations, we set the sample distance $\delta = 0.5$ in (6) and Algorithm 2, $\Gamma = 1$ for sampling candidate states, and $\Lambda = 1000$, $\gamma = 0.999$ in Algorithm 3. Figure 6 demonstrates the resulting shortest trajectories with respect to the given homology class. The 3-dimensional case is shown in Figure 7 with two obstacles.

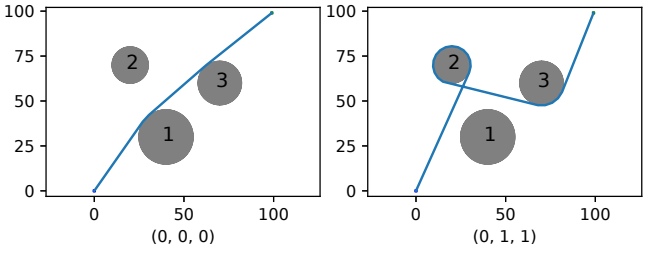

Fig. 6. Shortest path with respect to the given homology class. The labels of obstacles are marked on it and the target homology label (s_1, s_2, s_3) is shown below each figure.

The proposed method can be separated into two steps: graph building and then finding the shortest path using VIA. Given the graph, one can assign different states as the final states and then design the shortest path accordingly. Therefore, we demonstrate the efficiency of these two steps separately. During building the graph, there exists a trade-off between precision and efficiency. However, given a fixed number of obstacles, the number of $\mathcal{G}(\mathcal{V})$ will be squared with an increased number of sampled points. Fortunately, Algorithm 1 can also be paralleled and, hereby, dramatically decrease the computational time.

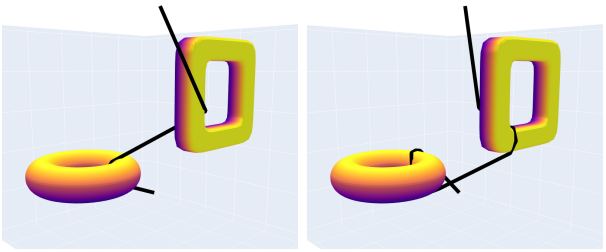

Fig. 7. Four trajectories start from $(0, 0, 0)$ to $(100, 100, 100)$ with the terminal homology class label $(s_1, s_2) = (1, 1)$ and $(-1, -1)$, respectively.

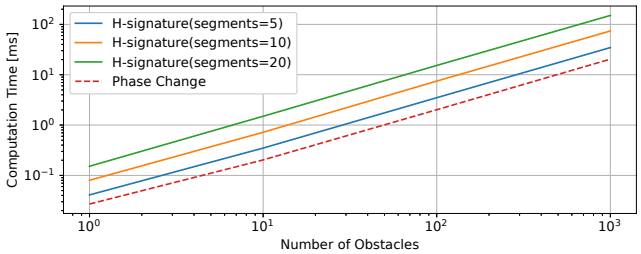

Fig. 8. Comparison of the computation time of getting the homology class label in a 3D environment with random obstacles between H-signature and the proposed phase-change-based method.

Figure 8 demonstrates the cost time of getting homology labels in 3D environments. H-signature methods are required to discretize the skeleton of obstacles to line segments, and the low number of segments may lead to inaccurate modeling and wrong labels. The proposed method can achieve fast labeling without discretizing obstacles. Figure 9 illustrates the computation time of the proposed method to find the shortest path from $(0, 0, 0)$ to $(1, 1, 1)$ with the homology class label $(1, \ldots, 1)$. Obstacles are randomly generated in the 3D space, and 100 locations are sampled for each obstacle. For parallel computing VIA using OpenMP, the 6-core Intel 9400F CPU is used. For the case using CUDA, the NVIDIA GTX 1660 GPU is leveraged. As the number of sampled locations increases linearly, the number of possible homology classes and the number of sampled states increase exponentially, therefore the calculation time also increases exponentially.

V. Conclusion

In this paper, we introduced a novel path-planning method with homology class constraints that can handle 2-dimensional and 3-dimensional environments in a unified manner. The idea of the proposed method is to first sample states around obstacles and connect them according to the phase change to build the graph. For 3-dimensional super-toroid obstacles, we defined the embedding space in which the phase change can be calculated in a manner similar to the case of 2-dimensional obstacles. Then Value Iteration Algorithm is leveraged to find the shortest path in the synthesized graph. The analysis of the embedding function suggests that even higher dimensional obstacles can also be mapped to 2-dimensional embedding space and therefore be processed by

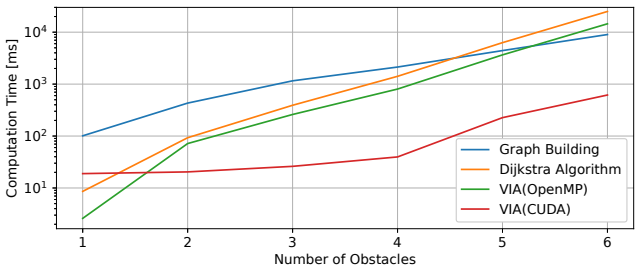

Fig. 9. Comparison of the computation time in a 3D environment with random obstacles between Dijkstra Algorithm and VIA. The complete computation time is the sum of the graph-building time and the searching time.

the proposed method if it has a proper formulation. This is planned to be investigated in future work. On the other hand, the proposed global searching method lacks efficiency when the number of possible homology classes increases, which is also planned to be addressed in the future.

References

[1] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, pp. 3–27, 2015.

[2] 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*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[4] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[5] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative LQR," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 244–254, 2019.

[6] K. Bergman and D. Axehill, "Combining homotopy methods and numerical optimal control to solve motion planning problems," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 347–354.

[7] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2612–2619.

[8] S. Bhattacharya, "Search-based path planning with homotopy class constraints," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 24, no. 1, 2010, pp. 1230–1237.

[9] W. He, Y. Huang, and S. Zeng, "Motion planning with homotopy class constraints via the auxiliary energy reduction technique," in *2022 American Control Conference, ACC 2022, Atlanta, GA, USA, June 8-10, 2022*. IEEE, 2022, pp. 4933–4938.

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

[11] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Optimal trajectory generation under homology class constraints," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 3157–3164.

[12] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.