Search-based Path Planning with Homotopy Class Constraints

Subhrajit Bhattacharya
Department of Mechanical Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104

Vijay Kumar
Department of Mechanical Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104

Maxim Likhachev
Department of Computer
and Information Science
University of Pennsylvania
Philadelphia, PA 19104 [*]

Abstract

Goal-directed path planning is one of the basic and widely studied problems in the field of mobile robotics. Homotopy classes of trajectories, arising due to the presence of obstacles, are defined as sets of trajectories that can be transformed into each other by gradual bending and stretching without colliding with obstacles. The problem of finding least-cost paths restricted to a specific homotopy class or finding least-cost paths that do not belong to certain homotopy classes arises frequently in such applications as predicting paths for dynamic entities and computing heuristics for path planning with dynamic constraints. In the present work, we develop a compact way of representing homotopy classes and propose an efficient method of graph search-based optimal path planning with constraints on homotopy classes. The method is based on representing the environment of the robot as a complex plane and making use of the Cauchy Integral Theorem. We prove optimality of the method and show its efficiency experimentally.

Introduction

Robot path planning is probably one of the most extensively studied problems areas in robotics (LaValle 2006). While there has been extensive research on path planning with a variety of constraints such as communication constraints (Abichandani, Benson, and Kam 2009), dynamics and environment constraints (Likhachev and Ferguson 2008) and time constraints (Hansen and Zhou 2007), research on path planning with homotopy constraints has been sparse (Grigoriev and Slissenko 1998; Schmitzberger et al. 2002).

A homotopy class of trajectories is defined by the set of trajectories joining same start and end points which can be smoothly deformed into one another without intersecting obstacles. Multiple homotopy classes arise due to presence of obstacles in the environment. Homotopy class constraints consist of restricting the solution to certain allowed homotopy classes or forbidding others.

Despite being mostly an uncharted research area, homotopy class constraints often appear in path planning problems. For example, in multi-agent planning problems (Zhang, Kumar, and Ostrowski 1998; Karabakal and Bean 1995), the trajectories often need to satisfy certain proximity or resource constraints or constraints arising due to tasks allocated to agents, which translates into restricting the solution trajectories to certain homotopy classes. In exploration and mapping problems (Bourgault et al. 2002), agents often need to plan trajectories based on their mission or part of the environment they are assigned for mapping or exploration, and hence restrict their trajectories to certain homotopy classes.

Motion planning with homotopy class constraints have been studied in the past using geometric approaches (Grigoriev and Slissenko 1998; Hershberger and Snoeyink 1991) and probabilistic road-map construction (Schmitzberger et al. 2002) techniques. Such techniques suffer from complexity of representation of homotopy classes and are not immediately integrable with standard graph search techniques. For example (Grigoriev and Slissenko 1998) describes homotopy classes by forming a *word* based on number of intersections of the trajectory with *cuts* of the obstacles, and consequently *reducing* the *word*. While comparing trajectories in different homotopy classes and finding the different homotopy classes in an environment is possible using such techniques, optimal path planning with homotopy class constraints is not achievable in an efficient way.

In the triangulation-based path planning (Demyen and Buro 2006) technique an environment with polygonal obstacles is first broken up into triangles by connecting the vertices of the polygons to create a *triangle graph*. This graph can be reduced to an *abstract triangle graph*, paths in which can represent various homotopy classes in the environment. A modified version of the A* search can then be performed in the graph (TRA*) to obtain a *channel of triangles*, from which a least-cost path can be derived using the Funnel algorithm (Kallmann 2005). However the construction of the triangulation graph relies on the assumption that the obstacles in the environment are polygonal, or at least can be approximated as polygons. In practical scenarios, for example when constructed from sensors on-board a robot, environments often contain oddly shaped as well as small obstacles and noise. The presence of small obstacles may create lots of thin triangles. Each disjoint obstacle needs to be considered to construct the abstract reduced graph in order to guarantee a collision-free path. Hence they contribute towards a

[*]We gratefully acknowledge support from ONR grant no. N00014-09-1-1052, NSF grant no. IIS-0427313, ARO grant no. W911NF-05-1-0219, ONR grants no. N00014-07-1-0829 and N00014-08-1-0696, and ARL grant no. W911NF-08-2-0004.

large number of homotopy classes, whereas we often may only need to consider a few large obstacles for defining homotopy classes. Moreover the cost function in triangulation-based path planning is restricted to the Euclidean length of the trajectory.

In this paper we propose a compact way of representing homotopy classes of trajectories which is independent of the geometry, discretization, cost function or search algorithm. Our method is based on Complex Analysis and exploits the Cauchy Integral theorem to characterize homotopy classes. We show that this representation can be seamlessly weaved into the standard graph search techniques in arbitrarily discretized environments and impose the desired homotopy class constraints. It is to be noted that the method we propose is independent of the discretization scheme of the environment, the nature of the cost function that needs to be optimized, or the search algorithm used. Hence this method can be incorporated into many existing planners, giving them the capability of imposing homotopy class constraints. Also, we can choose not to include certain obstacles if their sizes are too small for them to contribute towards creating homotopy classes. We have demonstrated our algorithm on different environments of various sizes with grid discretization as well as visibility graphs. Our experimental results demonstrate the efficiency, scalability and applicability of the method.

Complex Analysis

In this section we briefly review some of the fundamental theorems of Complex Analysis (Gamelin 2001).

The Cauchy Integral Theorem The Cauchy Integral Theorem states that if $f : \mathbb{C} \to \mathbb{C}$ is an Holomorphic (analytic) function in some simply connected region $\mathcal{R}$, and γ is a closed oriented (i.e. directed) contour completely contained in $\mathcal{R}$, then the following holds,

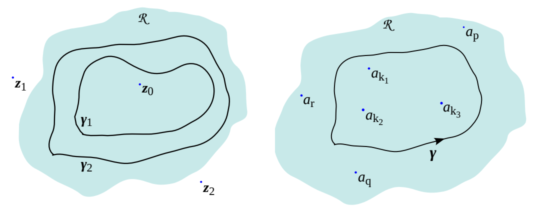
$$\oint_\gamma f(z)dz = 0 \tag{1}$$

Moreover, if z_0 is a point inside the region enclosed by γ, which has an anti-clockwise (or positive) orientation, and $F(z) = f(z)/(z - z_0)$ has a pole at z_0, then the following holds

$$\oint_\gamma \frac{f(z)dz}{z - z_0} = 2\pi i f(z_0) \tag{2}$$

The Residue Theorem A direct consequence of the Cauchy Integral Theorem, the Residue Theorem, states that, if $F : \mathcal{R} \to \mathbb{C}$ is a function defined in some simply connected region $\mathcal{R} \subseteq \mathbb{C}$ that has simple poles at the distinct points $a_1, a_2, \cdots, a_M \in \mathcal{R}$, and Holomorphic (analytic) everywhere else in $\mathcal{R}$, and say γ is a closed positively oriented Jordan curve completely contained in $\mathcal{R}$ and enclosing only the points $a_{k_1}, a_{k_1}, \cdots, a_{k_m}$ out of the poles of F, then the following holds,

$$\oint_\gamma F(z)dz = 2\pi i \sum_{l=1}^{m} \lim_{\xi \to a_{k_l}} (\xi - a_{k_l})F(\xi) \tag{3}$$



(a) The integrals over contours γ_1 and γ_2 are equal

(b) Only the poles enclosed by γ influence the value of the integral of F.

Figure 1: Cauchi Integral Theorem and Residue Theorem

The scenario is illustrated in Figure 1(b).

It is important to note that in both the Cauchy Integral Theorem and the Residue Theorem the value of the integrals are independent of the exact choice of the contour γ as long as the mentioned conditions are satisfied (see Figure 1(a)).

Representation of Homotopy Classes

Definition 1 (Homotopy Class of Trajectories). *Two trajectories (or paths), τ_1 and τ_2 are said to be in the same Homotopy Class iff one can be smoothly deformed into the other without intersecting obstacles. Otherwise they belong to different Homotopy classes.*

We represent the 2-dimensional configuration space of a robot by a complex plane, $\mathbb{C}$. The obstacles are assumed to be simply-connected regions in $\mathbb{C}$ and are represented by $\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_N$. We define one "representative point" per connected obstacle such that they lie in the interior of the respective obstacles. Thus we define the points $\zeta_i \in \mathcal{O}_i,\ \forall i = 1, \cdots, N$. Figure 2(a) shows such representative points inside three obstacles. As we will discuss later, it is not necessary that we choose representative points for all obstacles. We need to choose such points only on the larger and relevant obstacles that contribute towards the practical notion of homotopy classes. Smaller obstacles can be disregarded.

Definition 2 (Obstacle Marker Function). *For a given set of "representative points", we define the "Obstacle Marker Function" function $\mathcal{F}$ as follows,*

$$\mathcal{F}(z) = \frac{f_0(z)}{(z - \zeta_1)(z - \zeta_2)\cdots(z - \zeta_N)} \tag{4}$$

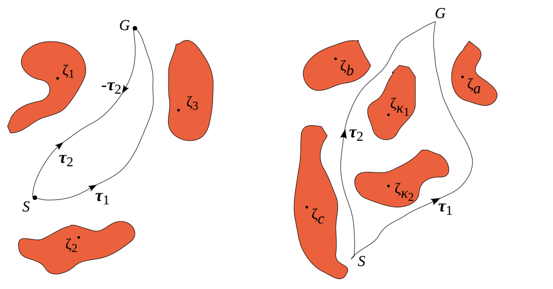
where f_0 is any analytic function over entire $\mathbb{C}$. Also, we define $\forall i = 1, \cdots, N$,

$$f_i(z) = \frac{f_0(z)}{(z - \zeta_1)\cdots(z - \zeta_{i-1})(z - \zeta_{i+1})\cdots(z - \zeta_N)} \tag{5}$$

Thus, $f_i(z) = (z - \zeta_i)\mathcal{F}(z)$, is analytic inside regions that do not contain $\zeta_1, \cdots, \zeta_{i-1}, \zeta_{i+1}, \cdots, \zeta_N$, but can contain ζ_i.

Assumption 1. We note that for a given set of obstacles we have significant amount of freedom in choosing the "representative points" inside the obstacles and the function f_0. We assume that our choice of $\zeta_1, \zeta_2, \cdots, \zeta_N$ and f_0 satisfies

$$\sum_{u \in S} f_u(\zeta_u) \neq 0 \tag{6}$$



(a) In same Homotopy class, forming a closed contour

(b) In different Homotopy classes, enclosing obstacles

Figure 2: Two trajectories in same and different homotopy classes

for any $S \subseteq \{1, 2, \cdots, N\}$. While it is evident that it is very unlikely that this condition will be violated in general case, we can still enforce the conditions by checking the chosen ζ_i's and f_0 for every S. In the unlikely event that (6) is not satisfied, we simply choose different ζ_i's and f_0.

Lemma 1. *If two trajectories τ_1 and τ_2 connecting the same points lie in the same homotopy class then the following holds,*

$$\int_{\tau_1} \mathcal{F}(z)dz = \int_{\tau_2} \mathcal{F}(z)dz \tag{7}$$

Proof. We note that by changing the orientation of a path over which an integration is being performed, we change the sign of the integral. If τ is a path, its oppositely oriented path is represented as $-\tau$. Thus, as we see from Figure 2(a), τ_1 along with $-\tau_2$ forms a positively oriented closed loop. Moreover, since τ_1 and τ_2 lie in the same homotopy class, the region enclosed by τ_1 and τ_2 does not contain any of the "representative points", ζ_i, hence rendering the function $\mathcal{F}$ analytic in that region. Hence from the Cauchy Integral Theorem we obtain,

$$\begin{aligned} & \int_{\tau_1} \mathcal{F}(z)dz + \int_{-\tau_2} \mathcal{F}(z)dz = 0 \\ \Rightarrow \quad & \int_{\tau_1} \mathcal{F}(z)dz - \int_{\tau_2} \mathcal{F}(z)dz = 0 \end{aligned}$$

Hence proved. □

Lemma 2. *If two trajectories τ_1 and τ_2 connecting the same points lie in different homotopy classes then the following holds,*

$$\int_{\tau_1} \mathcal{F}(z)dz \neq \int_{\tau_2} \mathcal{F}(z)dz \tag{8}$$

Proof. If τ_1 and τ_2 are in different homotopy classes, we can easily note that the closed positive contour formed by τ_1 and $-\tau_2$ will enclose one or more of the obstacles, and hence their corresponding "representative points". This is illustrated in Figure 2(b). Let us assume that enclosed "representative points" are $\zeta_{\kappa_1}, \zeta_{\kappa_2}, \cdots, \zeta_{\kappa_n}$. Moreover we note that the function $\mathcal{F}$ has only simple poles at $\zeta_1, \zeta_2, \cdots, \zeta_N$. Thus, by the Residue Theorem and Definition 2,

$$\begin{aligned} & \int_{\tau_1} \mathcal{F}(z)dz + \int_{-\tau_2} \mathcal{F}(z)dz \\ & \qquad = 2\pi i \sum_{u=1}^{n} \lim_{\xi \to \zeta_{\kappa_u}} (\xi - \zeta_{\kappa_u}) F(\xi) \\ \Rightarrow \quad & \int_{\tau_1} \mathcal{F}(z)dz - \int_{\tau_2} \mathcal{F}(z)dz = 2\pi i \sum_{l=u}^{n} f_{\kappa_u}(\zeta_{\kappa_u}) \\ & \qquad \neq 0, \\ & \qquad \text{by Assumption 1} \end{aligned}$$

Hence proved. □

Lemma 1 and 2 together imply that in a given environment, we can identify the homotopy classes of trajectories by the value of the line integration of the Obstacle Marker Function along the trajectory. For notational convenience for a given trajectory, τ, we write

$$L(\tau) = \int_{\tau} \mathcal{F}(z)dz \tag{9}$$

and call it the L-value of the trajectory τ. Thus,

Lemma 3. *The L-value of trajectories (connecting same two points) in same homotopy class are equal, while the L-value of trajectories in different homotopy classes are different.*

Search-based planning with Homotopy class constraints

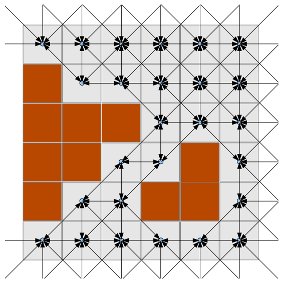
Our approach is to construct a graph by discretizing the environment and search it with any graph search algorithm such as Dijkstra's or A*.

The configuration space of a robot is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The nodes, of the graph represent coordinates, (x, y), which when represented as a complex number is $z = x + iy \in \mathcal{V}$. Inaccessible states (obstacles or states outside the workspace) do not constitute nodes. The directed edges represent possible transitions from one state to another and have weights or costs associated with each. Figure 3 illustrates a simple uniform discretization of the configuration space into an 8-connected grid, each cell representing a node of the Graph $\mathcal{G}$, and the connectivities are defined between neighboring cells. A path in the graph is equivalent to a trajectory.

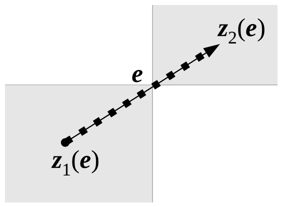
For a given graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for every edge $e = \{z_1 \to z_2\} \in \mathcal{E}$, assuming it to be a straight line segment, using the substitution $z = (1 - \lambda)z_1 + \lambda z_2$, one can write

$$\begin{aligned} L(e) &= \int_{e} \mathcal{F}(z)dz \\ &= \int_{0}^{1} \mathcal{F}\Big((1-\lambda)z_1 + \lambda z_2\Big)\,(z_2 - z_1)\,d\lambda \end{aligned} \tag{10}$$

This integral for every $e \in \mathcal{E}$ can be computed numerically for the graph $\mathcal{G}$ up to a desired numerical precision (Figure 3(b)). Alternatively, assuming e is small, one can use the following analytical formulation to compute $L(e)$.



(a) Graph formed by uniform discretization of configuration space and connecting each node with 8 neighbors



(b) An edge can be discretized further to compute $L(e)$

Figure 3: Discretized environment.

For computing the *L-value* of $e = \{z_1 \rightarrow z_2\}$ analytically, we assume that f_0 is chosen to be a polynomial of order $N-1$. Then $\mathcal{F}(z)$ can be written as the following *partial fraction decomposition*,

$$\mathcal{F}(z) = \sum_{l=1}^{N} \frac{A_l}{z - \zeta_l} \tag{11}$$

where, $A_l = \frac{f_0(\zeta_l)}{\prod_{\substack{j=1 \\ j \neq l}}^{N} (\zeta_l - \zeta_j)}$. Using some algebra and calculus in complex plane we get from (10),

$$\begin{aligned} L(e) &= (z_2 - z_1) \int_0^1 \sum_{l=1}^{N} \frac{A_l}{(z_2 - z_1)\lambda + (z_1 - \zeta_l)} d\lambda \\ &= \sum_{l=1}^{N} A_l \left(\ln(z_2 - \zeta_i) - \ln(z_1 - \zeta_l) \right) \end{aligned} \tag{12}$$

However we note that the *logarithm of a complex number* does not have an unique value. For any $z' \in \mathbb{C}$, $\ln(z') = \ln(|z'| e^{i(\arg(z') + 2k\pi)}) = \ln(|z'|) + i(\arg(z') + 2k\pi)$, $\forall k = 0, \pm 1, \pm 2, \ldots$ (where $\arg(x + iy) = \text{atan2}(y, x)$). Hence, following the assumption that e consists of a small line segment, we choose the *smallest* of all the possible values of $L(e)$ as follows,

$$\begin{aligned} L(e) = \sum_{l=1}^{N} A_l \Big[&\ln(|z_2 - \zeta_l|) - \ln(|z_1 - \zeta_l|) + \\ &i\Big(\arg(z_2 - \zeta_l) - \arg(z_1 - \zeta_l) + 2k_l\pi \Big) \Big] \end{aligned} \tag{13}$$

where, k_l is such that $|\arg(z_2 - \zeta_l) - \arg(z_1 - \zeta_l) + 2k_l\pi|$ is minimized. This corresponds to finding the minimum angle between the vectors $z_2 - \zeta_l$ and $z_1 - \zeta_l$.

The L-augmented Graph

Let $z_s = x_s + iy_s$ be the start coordinate and $z_g = x_g + iy_g$ be the goal coordinate. In the discussions that follow, by referring to the L-value of a state z, we will refer to the L-value of a trajectory connecting z_s to z. We note that the L-value of a state can assume only certain discrete complex numbers corresponding to the different homotopy classes of trajectories from z_s.

The Allowed and Blocked homotopy classes The planner is given one out of the two following sets of L-values for the state z_g (*i.e.* the L-value of trajectories joining z_s to z_g):

i. *The set of Blocked homotopy classes:* This is a set of L-values corresponding to the homotopy classes of trajectories that the solution is *not* allowed to be in. It is represented by $\mathcal{B} = \{\beta_1, \beta_2, \cdots, \beta_b\}$, where β_l is essentially a complex number representing the L-value of a homotopy class of trajectories that connect z_s and z_g. We note that $\mathcal{B}$ can be empty.

ii. *The set of Allowed homotopy classes:* This is a set of L-values corresponding to the homotopy classes of trajectories that the solution is allowed to be in. It is represented by $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_a\}$.

We note that $\mathcal{A}$ and $\mathcal{B}$ are essentially complement of each other and cannot have any element in common. Thus, if $\mathcal{U}$ is the set of all possible L-values of z_g, $\mathcal{B} = \mathcal{U}/\mathcal{A}$. Only one out of $\mathcal{A}$ and $\mathcal{B}$ needs to be explicitly defined.

The L-augmented Graph The identity of each state in $\mathcal{G}$ is its coordinate z. We modify the graph, $\mathcal{G}$, by appending the L-value of each state z to it (i.e. L-value of a trajectory from z_s to z). Thus a node in the Augmented Graph is given by $\{z, L(z_s \rightarrow z)\}$, where $z_s \rightarrow z$ represents a trajectory from z_s to z, and $L(z_s \rightarrow z)$ is its L-value. We note that the L-value appended to the state's identity indicates the homotopy class of a path found from z_s to z. For a given coordinate z, the L in the augmented state $\{z, L\}$ can assume discrete values corresponding to the homotopy classes of the trajectories joining z_s to z. Thus, we define the L-augmented graph of $\mathcal{G}$ as follows,

$$\mathcal{G}_L(\mathcal{G}) = \{\mathcal{V}_L, \mathcal{E}_L\}$$

where,

1. $$\mathcal{V}_L = \left\{ \{z, \Lambda\} \;\middle|\; \begin{array}{c} z \in \mathcal{V}, \text{ and,} \\ \Lambda \notin \mathcal{B} \text{ (or equivalently, } \Lambda' \in \mathcal{A}) \\ \text{if } z = z_g, \text{ and,} \\ \Lambda = L(z_s \rightarrow z) \text{ for some trajectory} \\ z_s \rightarrow z, \text{ from } z_s \text{ to } z \end{array} \right\}$$

2. And, edge $\{\{z, \Lambda\} \rightarrow \{z', \Lambda'\}\}$ is in $\mathcal{E}_L$ for $\{z, \Lambda\} \in \mathcal{V}_L$ and $\{z', \Lambda'\} \in \mathcal{V}_L$, iff
   i. $\{z \rightarrow z'\} \in \mathcal{E}$, and
   ii. $\Lambda' = \Lambda + L(z \rightarrow z')$, where $L(z \rightarrow z')$ is the L-value of the straight line segment joining the adjacent nodes z and z'

3. And, the cost/weight associated with an edge $\{\{z, \Lambda\} \rightarrow \{z', \Lambda'\}\} \in \mathcal{E}_L$ is same as the cost of the edge $\{z \rightarrow z'\} \in \mathcal{E}$.

The topology of this augmented graph is illustrated in Figure 4. A goal state z_g is the same in $\mathcal{G}$ irrespective for the path (τ_1 or τ_2) taken to reach it. Whereas in the L-augmented graph, the states are differentiated by the additional value of Λ. We can perform a graph search in the augmented graph, $\mathcal{G}_L$, using any standard graph search algorithm starting from the state $\{z_s, 0 + i0\}$. The goal state

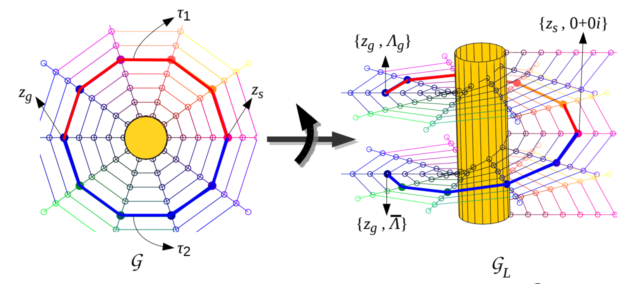


Figure 4: The topology of the augmented graph, $\mathcal{G}_L$ (right), compared againt $\mathcal{G}$ (left), for a cylindrically discretized configuration space around a circular obstacle

(i.e. the state, upon expansion of which we stop the graph search) is any of the states $\{z_g, \alpha\}$ for any $\alpha \in \mathcal{A}$ (or $\alpha \notin \mathcal{B}$ if $\mathcal{B}$ is provided instead of $\mathcal{A}$). We can use the same heuristic that we would have used for searching in $\mathcal{G}$, i.e. $h_L(z, \Lambda) = h(z)$. It is to be noted that $\mathcal{G}_L$ is essentially an infinite graph, even if $\mathcal{G}$ is finite. However the search algorithm needs to expand only a finite number of states. Since for a given z, the states $\{z, \Lambda\}$ can assume some discrete values of Λ (corresponding to the different homotopy classes), to determine if $\{z, \Lambda\}$ and $\{z, \overline{\Lambda}\}$ are the same states, we can simply compare the values of Λ and $\overline{\Lambda}$.

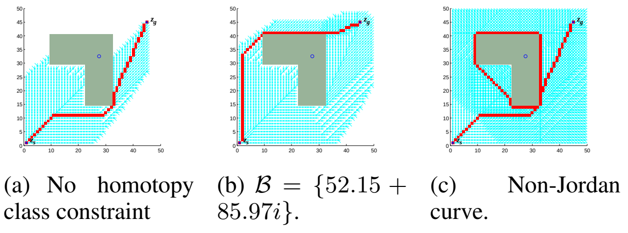
In our experiments we used both numerical (equation (10)) and analytical (equation (13)) approaches for computing the L-value of the edges. For the numerical integrations we chose 0.01 cell units as the step-size. For checking the equality of two L-values of a particular state, L_1 and L_2, we check if $||L_1 - L_2|| < \epsilon$ for some small ϵ. It is to be noted that for a particular state there are only discrete (generally widely separated) L-values corresponding to the discrete homotopy classes. Thus for all practical purposes we can safely keep ϵ quite high.

Theoretical Analysis

Theorem 1. *If $\mathcal{P}_L^* = \{\{z_1, \Lambda_1\}, \{z_2, \Lambda_2\}, \cdots, \{z_P, \Lambda_P\}\}$ is an optimal path in $\mathcal{G}_L$, then the path $\mathcal{P}^* = \{z_1, z_2, \cdots, z_P\}$ is an optimal path in the graph $\mathcal{G}$ satisfying the Homotopy class constraints specified by $\mathcal{A}$ and $\mathcal{B}$*

Sketch of Proof. By construction of $\mathcal{G}_L$, the path $\{z_1, z_2, \cdots, z_P\}$ (which is the projection of $\mathcal{P}_L^*$ on $\mathcal{G}$) satifies the given Homotopy class constraints. Moreover by definition, $\mathcal{P}_L^*$ is a minimum cost path in $\mathcal{G}_L$. Since the cost function in $\mathcal{G}_L$ is the same as the one in $\mathcal{G}$ and does not involve Λ, it follows that the projection of $\mathcal{P}_L^*$ on $\mathcal{G}$ given by $\mathcal{P}^* = \{z_1, z_2, \cdots, z_P\}$ is an optimal path in the graph $\mathcal{G}$ satisfying the Homotopy class constraints used to construct $\mathcal{G}_L$.

Illustrative Example Figure 5 demonstrates an example where we used a 50×50 uniformly discretized environment with a single obstacle (with $\zeta_1 = 27.5 + 32.5i$ marked with circle inside the obstacle). Each node in the graph, $\mathcal{G}$, is connected to its 8 neighbors. Thus the orientation of trajectory segments are constrained to multiples of $45°$ only. Start is at the bottom left point $(1, 1)$ in the environment, and the goal is on the top-right $(45, 45)$.



(a) No homotopy class constraint (b) $\mathcal{B} = \{52.15 + 85.97i\}$. (c) Non-Jordan curve.

Figure 5: Paths found and states expanded by A* search without and with homotopy class constraints

Figure 5(a) shows projection of the optimal path (shown in solid color) returned by A* search in $\mathcal{G}_L$ without any homotopy class constraint. The tree in *dashed color* shows the expanded states and their connection to the parent nodes. The L-value of the trajectory is found to be $52.15 + 85.97i$ and the total number of states expanded was 1011.

Next we block the homotopy class corresponding to L-value of $52.15 + 85.97i$ (i.e. set $\mathcal{B} = \{52.15 + 85.97i\}$ and plan the trajectory once again in $\mathcal{G}_L$. Thus we obtain a solution in a different homotopy class 5(b). The L-value of the new trajectory is found to be $-105.77 - 65.57i$ and the total number of states expanded was 1721.

As we can see from the figure, the blocking of a homotopy class made the search algorithm expand some extra states to find the solution in another homotopy class. However as we observe, the number of extra expansions is small. This is because during the exploration of states in the blocked homotopy class a large proportion of states in the desired homotopy class gets expanded as well.

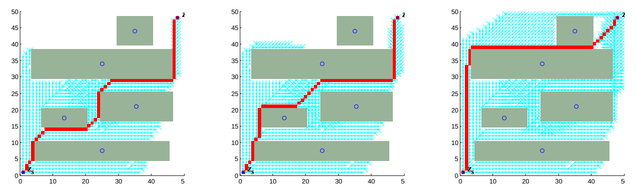
Also, it is to be noted that the exploration of the homotopy classes can be performed in a single run of graph search. This can be achieved by keeping on expanding the states in the L-augmented graph uninterruptedly, and noting down the states in the graph that correspond to the goal coordinates.

Figure 5(c) demonstrates what happens when we block both homotopy classes of trajectories that reach the goal from two sides of the obstacles. The trajectory we hence obtain is a non-Jordan curve (the one that intersects itself) that loops around the obstacle. If we keep searching while blocking one homotopy class after another, we will obtain least-cost trajectories in different homotopy classes in the order of their pathcosts. This includes non-Jordan curves.

Now imagine the scenario when we have a ζ_i set inside a very small obstacle near the start and far from the goal state. The expansion of states will result in a large number of loops around the small obstacle which will reduce the efficiency of the algorithm. Hence it's important that we choose the ζ_i carefully only inside relevant (large) obstacles which influence our notion of homotopy class of the trajectories.

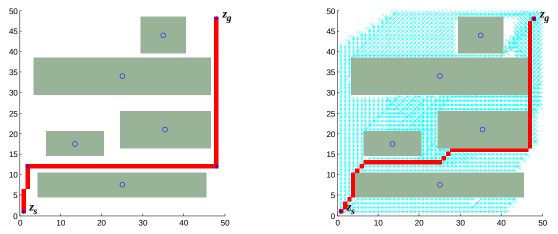
Experimental Analysis

In all of the following examples, except for the one with visibility graph, the state graph, $\mathcal{G}$, is generated by uniformly discretizing a rectangular configuration space and then connecting each node with its 8 neighboring nodes. The search algorithm used for searching in $\mathcal{G}_L$ was A*. The heuristic



(a) $\mathcal{B}$={} (b) $\mathcal{B}$= {−409.76+2557.70i}. (c) $\mathcal{B}$={−409.76+ 2557.70i,567.90+ 2220.77i}.

Figure 6: Exploring homotopy classes by blocking the class obtained from previous search



(a) Key-point generated trajectory. L-value $= 1530.94 + 531.55i$. (b) Optimal trajectory with $\mathcal{A} = \{1530.94{+}531.55i\}$.

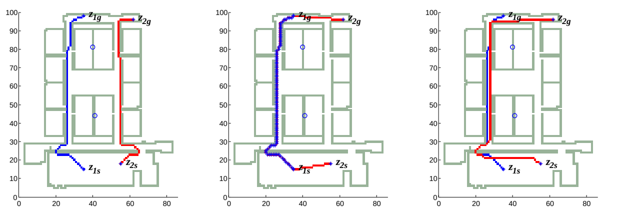
Figure 7: Homotopy class constraint determined using sub-optimal key-point generated trajectory.

used for the search was same as the heuristic for searching in $\mathcal{G}$. In our implementation the real and imaginary parts of L-values are floating point numbers forming a 2-element vector representing coordinates in the plane where we are planning the trajectories.

The functional form of f_0 was chosen as $f_0(z) = (z - BL)^a(z - TR)^b$, where BL and TR represent the complex coordinates of the bottom-left and the top-right points of the environment. We chose integer a and b such that $|a - b| \leq 1$ and $a + b = N - 1$. This lets us use the analytical formulation of (13) in computing $L(e)$. (But we have also tested the numerical integration of (10), and the results are identical except for longer run-time due to numerical integration). This functional form of f_0, which is analytic, is chosen so that the function $\mathcal{F}$ (and hence the L-values) scale well with the size of the environment and number of obstacles. Moreover such a function will almost always make sure that Assumption 1 holds. The ζ_i are chosen to be the points closest to the centroid of $\mathcal{O}_i$ and lying inside it (shown as circles in the figures 5-12).

Single robot trajectory planning

As described in the *illustrative example* section, Figures 5 and 6 demonstrate exploration of homotopy classes by blocking them one after another. For the example in Figures 6 (a), (b) and (c) the number of states expanded in the searches are 990, 1097 and 1622 respectively. Figure 7 demonstrates a similar example where we define homotopy classes using a sample (suboptimal) trajectory specified by key-points Figure 7(a). One can then compute the L-value for such a trajectory. It can then be used to search $\mathcal{G}_L$ for an optimal path in the same homotopy class (or different) as the sample trajectory (Figure 7(b)).



(a) Unconstrained plans of two robots (b) Robot 2 determines L-value of desired hmtp. class (c) Optimal plan with *visibility constraint* satisfied

Figure 8: 100 × 100 discretized environment with 2 *representative points* on the central large connected walls

Multiple robot visibility problem

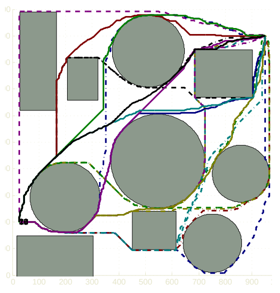
The problem of path planning for multiple robots with visibility constraints can also make use of our approach. If one robot needs to plan its path such that it is never obstructed from the view of another robot by some obstacle, we can apply the homotopy class technique to obtain the desired trajectories. In Figure 8(a)-(c) two robots plan trajectories to their respective goals. The robot on the right needs to plan a trajectory such that it is in the "visibility" of the robot on the left, whose trajectory is given. Thus, in order to determine the L-value of the desired homotopy class it first constructs a suboptimal path by connecting its own start and goal points to the start and goal of the left robot, such that the trajectory of the left robot is completely contained in it (Figure 8(b)) as key points. The L-value of this path gives the desired homotopy class, thus re-planning with that class as the only allowed class gives the desired optimal plan (Figure 8(c)).

Path prediction by homotopy class exploration

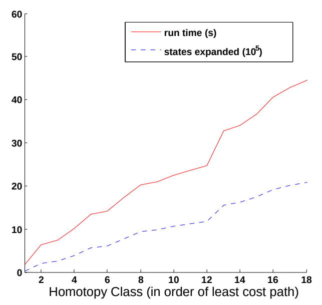
In order to demonstrate the scalability of our algorithm we constructed 10 large 1000 × 1000 environments using random circular and rectangular obstacles. The implementation was done in C++ running on an Intel Core 2 Duo processor with 2.1 GHz clock-speed and 4GB RAM. Figure 9(a) shows how for such an environment we can determine trajectories in different homotopy classes in order of their path costs. All the different trajectories in different homotopy classes were determined in a single run of graph search on $\mathcal{G}_L$. Figure 9(b) and Table 9(c) demonstrate the efficiency of the searches. The time indicates the cumulative time during the search until a shortest-path trajectory in a particular homotopy class is found. This is relevant to problems of tracking dynamic entities, such as people, where one often needs to predict possible paths in order to bias the tracker or to deal with occlusion by anticipating where the dynamic entity will appear. Since people can choose different paths to their destinations, we need to be able to predict least cost paths that lie in different homotopy classes.

Arbitrary cost functions

Our method is not limited to Euclidean length cost functions. It can deal with arbitrary cost functions. For example, in Figure 10 there are two large obstacles and a *communication base* to the left of the environment marked by the bold dotted line, $x = 0$. An agent is supposed to plan its path from the bottom to the top of the environment, while minimizing a weighted sum of the length of the trajectory and



(a) Paths in 20 different homotopy classes

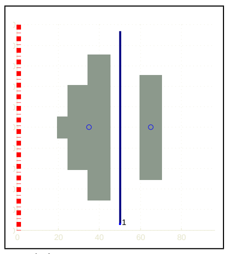


(b) Run-time & states expanded for finding the least-cost paths in a particular run

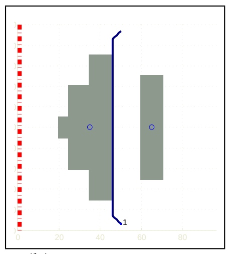
Hmtp. class (i)	time ellapsed until i^{th} hmtp. class explored (s)			states expanded cumulative (10^6)		
	min	max	mean	min	max	mean
1	1.41	2.01	1.71	0.021	0.039	0.032
2	3.58	8.58	5.15	0.099	0.313	0.170
3	5.09	9.69	6.77	0.180	0.375	0.244
4	6.13	12.46	8.92	0.237	0.494	0.345
5	7.80	17.74	11.50	0.285	0.776	0.472
6	10.53	18.56	13.05	0.422	0.825	0.555
7	10.92	19.74	15.38	0.473	0.888	0.681
8	13.35	20.32	17.01	0.604	0.935	0.773
9	15.08	21.76	18.60	0.693	1.027	0.858
10	15.53	26.28	20.87	0.720	1.252	0.978

(c) Statistics of searching least-cost paths in first 10 homotopy classes in 10 randomly generated environments. The numbers represent the cumulative values till the i^{th} homotopy class is explored.

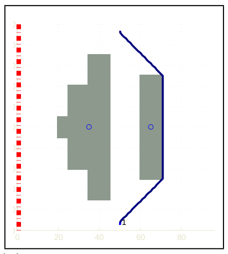
Figure 9: Exploring homotopy classes in 1000×1000 discretized environments to find least cost paths in each



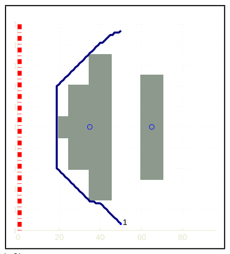
(a) $w = 0.0, \mathcal{B} = \{\}$



(b) $w = 0.01, \mathcal{B} = \{\}$



(c) $w = 0.0, \mathcal{B} = \{-8.41 + 8.41i\}$



(d) $w = 0.01, \mathcal{B} = \{-8.41 + 8.41i\}$

Figure 10: Planning with non-Euclidean length as cost as well as homotopy class constraint

the distance of the trajectory from the *communication base*. Thus, in this case, besides the transition costs of the states in $\mathcal{G}$, each state, $z = x + iy \in \mathcal{G}$, is assigned a cost $w \cdot x$, the penalty on separation from the *communication base*. Thus the net penalized cost of the trajectory, τ, that is being minimized is of the form $c = \int_\tau ds + w \int_\tau x(s)ds$, where x is the x-coordinate of the points on the trajectory, parametrized by s, the length of the trajectory. The trajectories in figures 10(a) and (b) with penalty weights $w = 0$ and $w = 0.01$ respectively have L-values of $-8.41 + 8.41i$. Blocking this homotopy class, but having a small penalty over distance from *communication base* gives the trajectory in 10(d) that passes close to the *communication base*.

Planning with additional coordinates

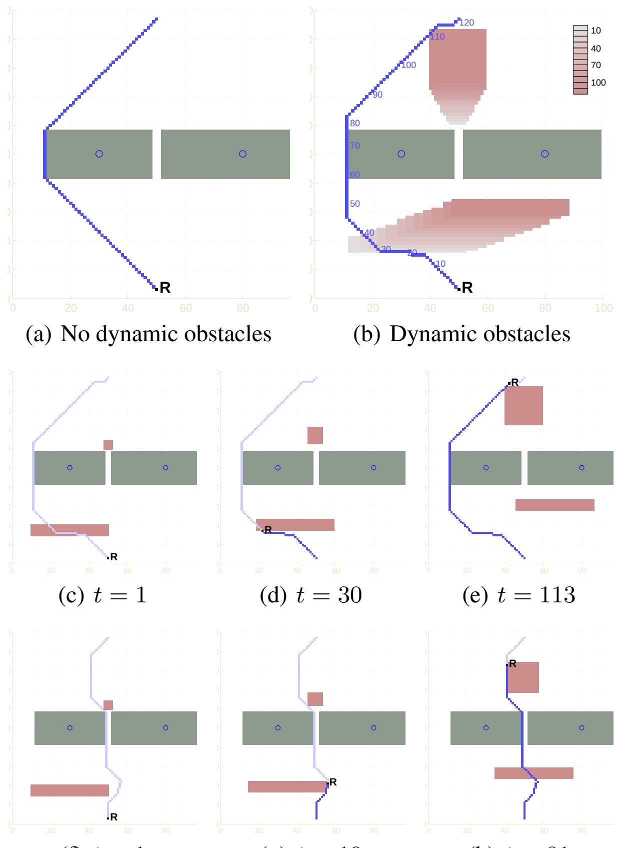
In Figure 11, besides x and y, we have used $time$ as a third coordinate in $\mathcal{G}$. There are two dynamic obstacles in the environment - the one at the bottom only translates, while one near the top both translates as well as expands in size. We have two *representative points* on the two static obstacles. Figure 11(a) shows the solution upon blocking the first homotopy class in the environment without the dynamic obstacles. Figure 11(b) shows the planned trajectory in the environment with dynamic obstacles (with the color intensity representing the time coordinate). Figure 11(c-e) show the execution of the trajectory at different instants of time of the same. Figure 11(g-h) show the execution of the plan without homotopy class constraint.

Implementation using Visibility Graph

To demonstrate the versatility of the proposed algorithm we implemented it using a Visibility Graph as the state graph, $\mathcal{G}$. Figure 12 shows the visibility graph generated in an environment with polygonal obstacles and the shortest paths in the first 9 homotopy classes. Obstacles were *inflated* in order to incorporate collision safety and circular obstacles were approximated by polygons. *Representative points* were placed only on the large obstacles (determined by threshold on diameter and marked by blue circles in the figure) and visibility graph was constructed. A* search was used for searching the L-augmented graph. The implementation was made in MATLAB. The average run-time of the search until the 9^{th} homotopy class was explored was 0.4 seconds and about 100 states were expanded.

Conclusion

In this paper we have proposed an efficient way of representing homotopy classes of trajectories using the line integral of "Obstacle Marker Function" over the trajectory in a complex plane. Using the proposed representation, we have shown that homotopy class constraints can be directly weaved with graph search techniques for determining optimal path constrained to certain homotopy classes or forbidden from others. We have proved the optimality of the method and have experimentally demonstrated its efficiency, versatility and several applications.


(a) No dynamic obstacles (b) Dynamic obstacles

(c) $t = 1$ (d) $t = 30$ (e) $t = 113$

(f) $t = 1$ (g) $t = 19$ (h) $t = 81$

Figure 11: Planning with time as an additional coordinate. The postion of the agent is denoted by **R** in figures (c)-(h).

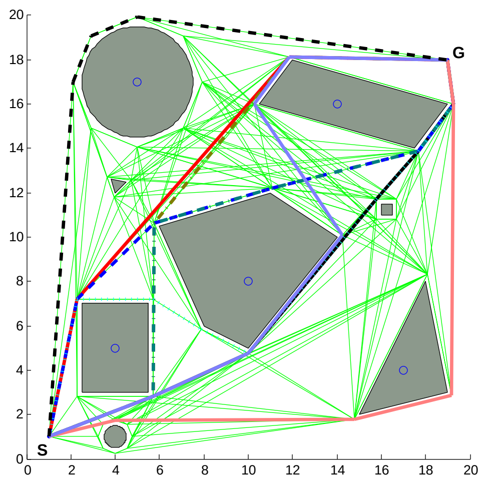


Figure 12: Exploring homotopy classes using a Visibility Graph

References

Abichandani, P.; Benson, H. Y.; and Kam, M. 2009. Multi-vehicle path coordination in support of communication. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, 3839–3846. Institute of Electrical and Electronics Engineers Inc., The.

Bourgault, F.; Makarenko, A. A.; Williams, S. B.; Grocholsky, B.; and Durrant-Whyte, H. F. 2002. Information based adaptive robotic exploration. In *in Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, 540–545.

Demyen, D., and Buro, M. 2006. Efficient triangulation-based pathfinding. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, 942–947. AAAI Press.

Gamelin, T. W. 2001. *Complex analysis*. Springer Science.

Grigoriev, D., and Slissenko, A. 1998. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In *ISSAC '98: Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, 17–24. New York, NY, USA: ACM.

Hansen, E. A., and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 28:267–297.

Hershberger, J., and Snoeyink, J. 1991. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl* 4:331–342.

Kallmann, M. 2005. Path planning in triangulations. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*.

Karabakal, N., and Bean, J. C. 1995. A multiplier adjustment method for multiple shortest path problem. Technical report, The University of Michigan.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

Likhachev, M., and Ferguson, D. 2008. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of Robotics: Science and Systems (RSS)*.

Schmitzberger, E.; Bouchet, J.; Dufaut, M.; Wolf, D.; and Husson, R. 2002. Capture of homotopy classes with probabilistic road map. In *International Conference on Intelligent Robots and Systems*, volume 3, 2317–2322.

Zhang, H.; Kumar, V.; and Ostrowski, J. 1998. Motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*. Leuven, Belgium: IEEE.