THE SHORTEST PATH AMID 3-D POLYHEDRAL OBSTACLES

SHUI-NEE CHOW, JUN LU, HAO-MIN ZHOU

Abstract. It is well known that the problem of finding the shortest path amid 3-D polyhedral obstacles is a NP-Hard problem. In this paper, we propose an efficient algorithm to find the globally shortest path by solving stochastic differential equations (SDEs). The main idea is based on the simple structure of the shortest path, namely it consists of straight line segments connected by junctions on the edges of the polyhedral obstacles. Thus, finding the shortest path is equivalent to determining the junctions points. This reduces the originally infinite dimensional problem to a finite dimensional one. We use the gradient descent method in conjunction with Intermittent Diffusion (ID), a global optimization strategy, to deduce SDEs for the globally optimal solution. Compared to the existing methods, our algorithm is efficient, easier to implement, and able to obtain the solution with any desirable precisions.

1. Introduction

Finding the shortest path in the presence of obstacles is one of the fundamental problems in path planning and robotics. The problem can be described as follows: given a finite number of obstacles in $\mathbf{R}^2$ or $\mathbf{R}^3$, what is the shortest path connecting two given points X, Y while avoiding the obstacles. The problem has received great attention during the last few decades, and many techniques have been developed for polygonal obstacles in $\mathbf{R}^2$, where the problem can be reformulated as an optimization problem on a graph, and therefore can be solved by combinatorial methods. For example, by using the shortest path map method, Hershberger and Suri [6] found an optimal $\mathbf{O}(n \log n)$ polynomial time algorithm where n is the total number of vertices of all polygonal obstacles. We refer to [8, 9] for a survey of the results and references therein. However, Canny and Rief [1] proved that this problem in $\mathbf{R}^3$ becomes NP-hard under the framework known as "configuration space". This is mainly because the shortest path doesn't necessarily pass through the set of vertices of polyhedrons. Instead, it may go through the interior points of edges, and this makes the optimal algorithm in 2-D fail.

Two different approaches were developed later to overcome this difficulty. One is to find a path that is $1 + \epsilon$ times the length of the shortest one. The idea is to subdivide the edges in certain ways and adopt the same optimal combinatorial methods which are effective in $\mathbf{R}^2$. Following this idea, Papadimitriou developed an algorithm in [10] with complexity $O(\frac{1}{\epsilon})$. In a special case where the shortest path is unique, one can define the precision δ of the problem, which is the difference between the shortest path and the second shortest path. Given $\epsilon < \delta$, a faster algorithm was developed in [2] with complexity $O(\log(\frac{1}{\epsilon}) + P(1/\delta))$ for some polynomial P. The idea is to apply the appoach in [10] to obtain a good initialization within error δ to the shortest path, and then use a gradient descent strategy to improve the accuracy to ϵ.

This work is partially supported by NSF Faculty Early Career Development (CAREER) Award DMS-0645266 and DMS-1042998, and ONR Award N000141310408.

Another commonly used approach divides the problem into two parts: (i) find the sequence of edges that the shortest path may go through, and (ii) find the optimal connecting points on those edges. For convex polyhedral obstacles, it is observed in [13] that the total number of possible sequences are of order $O(n^{7k}k^k)$ where n is the total number of vertices and k is the number of obstacles. Part (ii) is proven to be NP-hard [5]. A different method, called unfolding technique, was introduced in [11] under a theoretical computation model in which it assumes any infinite-precision real arithmetic operation requires constant time. However, this assumption may not be practical.

On the other hand, several PDEs based methods have been proposed to tackle the shortest path problem with obstacles having smooth boundaries. For example, path evolution method finds the solution by solving a 2-point boundary value ordinary differential equation (ODE), resulting local optimal solutions. The front propagation method finds the global solution by solving an eikonal equation, a partial differential equation (PDE). The numerical solution of the eikonal equation can be computed by fast marching [12] or fast sweeping method [14].

In [4], we proposed a different algorithm called Evolving Junctions on Obstacle Boundaries (E-JOB) for finding the shortest path. E-JOB is a general framework which can be applied to environment with obstacles of arbitrary shape (continuous or discrete) in any dimension ($\mathbf{R}^2, \mathbf{R}^3$ or higher). The key idea is dimension reduction. It takes advantage of a simple geometric structure of the shortest path, i.e. the shortest path is composed by line segments and arcs on the obstacle boundaries. The shortest path is determined completely by the junctions of those segments. In this way, the problem becomes how to find those junction points on the boundaries. In other words, the original infinite dimensional problem of finding the whole path is converted to a finite dimensional problem of finding only the junction points. The optimal position of those junctions can be determined efficiently by the gradient descent method. To avoid the problem of stuck in local minimizers associated, a global optimization strategy called intermittent diffusion (ID) is adopted. This strategy adds random perturbations to the ODEs of the gradient descent method in a temporal discontinuous fashion, which leads to stochastic differential equations (SDEs). It obtains the globally shortest path with probability $1 - \delta$ where δ is an arbitrarily small number.

In this paper, we focus on applying E-JOB to the shortest path problem with polyhedral obstacles in $\mathbf{R}^3$. The restriction on polyhedral obstacles allows us to achieve further dimension reductions. For obstacles with smooth boundaries, the implementation of E-JOB requires computations of geodesics on the boundaries between two given points. In [4], this is achieved by either traversing the boundaries in $\mathbf{R}^2$, or fast marching on the boundary surfaces in $\mathbf{R}^3$. However, for polyhedral obstacles, the geodesics also has a similar simple structure, i.e. the geodesic between two points on a polyhedron is a conjunction of line segments whose ending points are located on polyhedron edges. And to determine the geodesic is equivalent to determine those junction points. Therefore the overall shortest path connecting X and Y is merely a conjunction of line segments whose ending points lie on obstacle edges. In other words, the domain for each junction is an 1-D interval. This makes the algorithm extremely simple and efficient.

A feature of this study is that we do not restrict the obstacles to be convex polyhedrons. The algorithm we develop can equally be applied to non-convex polyhedrons. For polyhedrons with Euler characteristic 2, which include all convex polyhedrons and concave polyhedrons without holes, our algorithm can find the globally optimal path with probability arbitrarily close to 1 in a finite time. However, when dealing

with more sophisticated polyhedrons, for example, polyhedrons with complicated holes, certain topological problems emerge, and prevent us from obtaining the globally optimal path. We will discuss this issue at the end of the paper as well as some possible solutions.

It should be noted that our approach resembles the one in [2] in the sense that both employ a gradient descent strategy. However, before the strategy can be applied, the method in [2] requires the assumption that the shortest path is unique, and an initialization that approximates the shortest path within error δ (precision) to start with (achieved by using [10]). Our approach needs neither of them. In fact, our approach can be viewed as a way to find both the edge sequence and the optimal connecting points in a unified manner, thanks to the introduction of randomness into the differential equations. Below, we summarize some advantages of our algorithm:

(1) The algorithm can obtain the shortest path in any precision. This is because only a system of SDEs needs to be solved which involves no subdivision of edges.
(2) The algorithm is able to handle non-convex polyhedral obstacles.
(3) The algorithm is easy to implement.
(4) The algorithm is fast. Since we solve an initial value problem of SDEs, the results can be obtained efficiently by various established schemes.

The paper is arranged as follows. In Section 2, we give the derivation of the algorithm following the ideas presented in [4]. The algorithm is then presented whose details follows afterwards. In Section 3, we give several interesting examples. Finally, we discuss the topological issues arosen when dealing with polyhedrons with holes.

2. New Algorithm

In this section, we present our new algorithm for the shortest path problem with polyhedron obstacles. We start with some mathematical description of the problem, through which we introduce notations needed in the rest of the paper. The algorithm follows afterwards and its details are presented at the end of this section.

Let $\{P_k\}_{k=1}^N$ be N polyhedral obstacles in $\mathbf{R}^3$. Each obstacle P_k is determined uniquely by its vertices, edges and faces. Denote V, E, F the set of vertices, edges and faces of P_k respectively. We do not limit the polyhedrons to be convex. However, we will focus on polyhedrons without holes in this section, i.e. polyhedrons whose Euler characteristic is 2. The Euler characteristic is defined by

$$\chi = |V| - |E| + |F|.$$

Polyhedrons with holes will be discussed in the last section. For any edge $e \in E$, it has a representation

$$e = (\mathbf{u}, \mathbf{v})$$

where $\mathbf{u}, \mathbf{v}$ are the coordinates of the ending points of e. Any point x on edge $e = (\mathbf{u}, \mathbf{v})$ can then be represented by the following expression

$$x(\mathbf{u}, \mathbf{v}, \theta) = \theta\mathbf{u} + (1-\theta)\mathbf{v}. \tag{1}$$

Thus to determine the position of a point on an edge, one only needs to find its corresponding θ.

2.1. Geodesics on polyhedrons.

For any two points x, y on the edges of P_k, we can define the distance $d_k(x, y)$ between them to be the length of the shortest path on P_k connecting x and y. If we view P_k as a surface in $\mathbf{R}^3$, i.e. a two dimensional manifold, then $d_k(x, y)$ is nothing but the geodesic on P_k connecting x and y. For instance, for any x and y in a tetrahedron, $d(x, y) = \|x-y\|$ since the line

segment joining them is on the surface. For general polyhedrons, the shortest path is composed by a sequence of line segments connected to each other. To be more specific, the shortest path can be represented by $(x_0, x_1, x_2, \ldots, x_{n_k}, x_{n_k+1})$ where $x_0 = x, x_{n_k+1} = y$ and each x_i is a point on some edge $e_i = (\mathbf{u}_i, \mathbf{v}_i)$. The shortest distance $d_k(x, y)$ therefore equals

$$d_k(x, y) = \mathcal{L}(x_1, x_2, \ldots, x_{n_k}) = \sum_{i=0}^{n_k} \|x_{i+1} - x_i\|.$$

Denote $x_i = \theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i$, we then have

$$\mathcal{L}(\theta_1, \ldots, \theta_{n_k}) = \mathcal{L}(x_1, \ldots, x_{n_k}) = \sum_{i=0}^{n_k} \|\theta_{i+1}\mathbf{u}_{i+1} + (1-\theta_{i+1})\mathbf{v}_{i+1} - \theta_i \mathbf{u}_i - (1-\theta_i)\mathbf{v}_i\|.$$

2.2. **Structure of the shortest path.** A path is a curve $\gamma \in \mathbf{R}^3$, which is a continuous map

$$\gamma(\cdot)\colon [0, 1] \to \mathbf{R}^3.$$

We denote $L(\gamma)$ the Euclidean length of the path γ. We are concerned with all feasible paths $\mathbf{F}$, i.e. paths that don's intersect with any obstacle P_k. The shortest path connecting X and Y is then given by

$$\gamma_{opt} = \text{argmin}_{\gamma \in \mathbf{F}} L(\gamma).$$

In [4], we proved that the shortest path has a simple structure, i.e. it is composed by line segments outside the obstacles and paths on the boundary of the obstacles. Since all the obstacles here are polyhedrons, the paths on the boundaries of the obstacles also consist of a sequence of line segments connected by points on the edges. Therefore, by putting all the connecting points together and relabeling them, the shortest path connecting X and Y can be represented by $(x_0, x_1, x_2, \ldots, x_n, x_{n+1})$ where $x_0 = X, x_{n+1} = Y$.

Let us denote

$$J(x_i) = \|x_{i-1} - x_i\| + \|x_{i+1} - x_i\|. \tag{2}$$

Then the length of the path is

$$\mathcal{L}(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} J(x_i). \tag{3}$$

Again all x_is are on the edges of the obstacle. Denote $x_i = \theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i$, $J(x_i)$ then becomes

$$J(\theta_i) = \|\theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i - x_{i-1}\| + \|\theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i - x_{i+1}\|. \tag{4}$$

2.3. **Optimal Path.** To find the optimal path, we differentiate $J(\theta_i)$ with respect to θ_i to obtain

$$\nabla J(\theta_i) = \frac{(x_i - x_{i-1}) \cdot (\mathbf{u}_i - \mathbf{v}_i)}{\|x_i - x_{i-1}\|} + \frac{(x_i - x_{i+1}) \cdot (\mathbf{u}_i - \mathbf{v}_i)}{\|x_i - x_{i+1}\|}. \tag{5}$$

So using the method of gradient decent, we can find the optimal position θ_i following a system of ODEs,

$$\frac{d\theta_i}{dt} = -\nabla J(\theta_i). \tag{6}$$

In order to find the global optimal path, we adopt a strategy called Intermittent Diffusion, i.e. we evolve the following stochastic differential equation

$$\frac{d\theta_i}{dt} = -\nabla J(\theta_i) + \sigma(t) dW(t) \tag{7}$$

where $\sigma(t)$ is a step function and $W(t)$ is standard Brownian motion. More precisely,

$$\sigma(t) = \sum_{l=1}^{m} \sigma_l \mathbf{1}_{[S_l,T_l]}(t) \tag{8}$$

with $0 = S_1 < T_1 < S_2 < T_2 < \cdots < S_m < T_m < S_{m+1} = T$, and $\mathbf{1}_{[S_l,T_l]}$ being the indicator function of interval $[S_l, T_l]$. Here S_l, T_l, σ_l are chosen to be random and the magnitude of σ_l depends on the size of the obstacles. For more details, see [3].

Theorem 1. *If all the obstacles have Euler characteristic 2, then for any small number ϵ,equation* (7) *converges to the global minimizer with probability $1 - \epsilon$ for suitable $\sigma(t)$.*

We will postpone the proof until the last section when we discuss the topological issues.

2.4. **Numerical Scheme.** In this section, we discuss how to solve equation (7).

2.4.1. *Discretization of SDE.* We use forward Euler method to discretize equation (7) as following

$$\frac{\theta_i^{j+1} - \theta_i^j}{\Delta t} = -\nabla J(\theta_i^j) + \sigma(j\Delta t)\sqrt{\Delta t}\xi$$

where $\xi \sim N(0, 1)$ is a normal random variable.

2.4.2. *Initialization.* We can use the optimal path whose junctions are restricted on vertices of the obstacles to initialize the path. This initialization can be obtained efficiently by a method called visibility graph . The visibility graph W is a weighted graph whose nodes are the vertices of all the obstacles as well as the starting and ending points X, Y, and there is an edge between vertices $\mathbf{u} \in W$ and $\mathbf{v} \in W$ if and only if they are visible to each other, that is, if the line segment $\overline{\mathbf{uv}}$ doesn't intersect with any obstacles. The weight of edge $\mathbf{uv}$ is simply the Euclidean distance of $\overline{\mathbf{uv}}$. One thing to notice is that the visibility graph we construct here is essentially 2D, in the sense that it encodes whether two points are visible to each other. This is fundamentally different from the 3D reduced visibility graph (3DRVG) in the introduction. 3DRVG consists of connected planes as opposed to straight line segments which becomes complicated when there are more than one obstacles. See [7]. After the visibility graph is constructed, the initialization is the shortest path between X and Y on the visibility graph W which can be obtained efficiently by Dijkstra algorithm..

2.4.3. *Evolution when a junction reaches a vertex.* In the proposed method, the junctions move according to the SDEs if they are on the interior of edges. When a junction $x = (\mathbf{u}, \mathbf{v}, \theta)$ reaches to a vertex $\mathbf{u}$ following the gradient flow, it continues moving according to different rules depending on whether the two neighbors of x are on the same obstacle or not. If the neighbors of x are both on the same obstacle as x, we call x an interior junction, otherwise we call x an exterior junction. In other words, an exterior junction is one of the two ending points of the line segments that connects two different obstacles. The following are the rules for interior and exterior junctions reaching the vertices respectively.

Case 1. $x = (\mathbf{u}, \mathbf{v}, \theta = 1)$ is an interior junction. See the following illustration where (x_1, x_2, x, x_4) is the path on the obstacle and x_1, x_4 are exterior junctions. When x hits $\mathbf{u}$ ($\theta = 1$), path $(x_1, u = x, x_4)$ will have smaller length than $(x_1, x_2, u = x, x_4)$ length. In other words, all the junctions adjacent to $\mathbf{u}$ will be dragged to $\mathbf{u}$ except the exterior junctions. Hence we remove all the

junctions adjacent to $\mathbf{u}$ and add junctions on the edges adjacent to $\mathbf{u}$ that haven't been occupied which results in a new path (x_1, x_6, x_5, x_4).

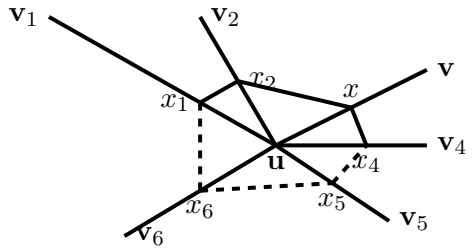


FIGURE 1. Movement of interior junction

Case 2. $x = (\mathbf{u}, \mathbf{v}, \theta = 1)$ is an exterior junction. Let z be its neighbor on the same obstacle and y be the neighbor on another obstacle. x will move to a different feasible edge $\mathbf{uw}$ once it hits $\mathbf{u}$. Edge $\mathbf{uw}$ is said to be feasible if

(a) The line segment joining y and $\mathbf{u} + \Delta\theta(\mathbf{w} - \mathbf{u})$ doesn't intersect with any obstacle for arbitrarily small $\Delta\theta$.
(b) The length $J(x)$ decreases as x moves away from $\mathbf{u}$ on $\mathbf{uw}$, i.e.

$$\nabla J(x) \cdot (\mathbf{w} - \mathbf{u}) < 0.$$

We collect all the feasible directions and select one of them with equal possibility, x then continues evolving according to the flow. Depending on whether neighbor z is visible to edge $\mathbf{uw}$, the new path are as follows:

i. z is on the same face as $\mathbf{uw}$, then the new path becomes $(\cdots, y, x', z, \cdots)$ where $x' \in \mathbf{uw}$.
ii. z is not on the same face as $\mathbf{uw}$, then x is used as an intermittent junction and the new path becomes $(\cdots, y, x', x, z, \cdots)$ where $x' \in \mathbf{uw}$.

For an illustration, see the following example. The feasible direction are $\mathbf{uv}_8$ and $\mathbf{uv}_2$. z is visible to $\mathbf{uv}_8$, the path after evolution is simply (y, x', z). On the other hand, z is invisible to $\mathbf{uv}_2$, the path after evolution is simply (y, x', x, z).

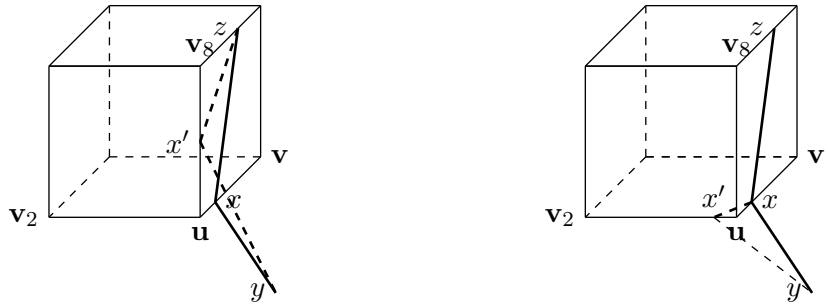


FIGURE 2. Movement of exterior junction x. The left figure corresponds to case (i) and the right corresponds to case (ii)

2.4.4. *Add and remove junctions.* During the evolution of each point, we may need to add or eliminate junction points. When two neighboring junctions x, y are both exterior and $\overline{xy}$ intersect with obstacle $P_{k_1}, P_{k_2}, \cdots, P_{k_r}$, we initialize a path with x, y being the starting and ending points and $\{P_{k_i}\}_{i=1}^r$ being the obstacles. Denote the new added junctions by $(x_{n+1}, x_{n+2}, \cdots, x_{n+s})$ where s is the total number of

new junctions. Then they are inserted into the set of junctions in order and the evolution process continues. On the other hand, when two neighboring junctions x, y are both exterior and x meets y, we may shorten the path by removing x and y. More precisely, let z_1 be the other neighbor of x and z_2 be the other neighbor of y, i.e. the path contains $(\cdots, z_1, x, y, z_2, \cdots)$ as a fraction. Since $x = y$, we may connect z_1 and z_2 directly which shorten the length. In other words, we have the new fraction $(\cdots, z_1, z_2, \cdots)$. Notice, the line segment $\overline{z_1 z_2}$ may intersect with some obstacles. Again we add the necessary junctions as described above.

2.5. **Algorithm.** We present our algorithm below

Input: number of intermittent diffusion intervals m, duration of diffusion $\Delta T_l, l \leq m$. diffusion coefficients $\sigma_l, l \leq m$.
Output: The optimal set U_{opt} of junctions.

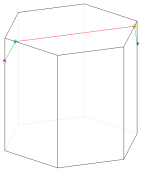
Initialization. Find the initial set U of junction points.
for $l = 1 : m$
  $U_l = U$;
  for $x_i = (\mathbf{u}, \mathbf{v}, \theta_i^0) \in U_l$
    for $j = 1 : \Delta T_l$
      Update x according to (7), i.e. $\theta_i^{j+1} = \theta_i^j + (-\nabla J(\theta_i^j) + \sigma_l \sqrt{\Delta t}\xi)\Delta t$;
      Update set U_l, i.e. remove junctions from or add junctions to U_l;
    end
    while $|\theta_{i+1}^{j+1} - \theta_i^j| > \epsilon$ (or other convergence criterion)
      Update x according to (6), i.e. $\theta_i^{j+1} = \theta_i^j - \nabla J(\theta_i^j)\Delta t$;
      Update set U_l;
    end
  end
end
Compare U_ls and set $U_{opt} = \operatorname{argmin}_{l \leq m} \mathcal{L}(U_l)$.

3. Numerical Examples

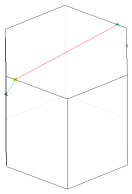
We show several examples in this section to illustrate the paths obtained by our algorithm. The diffusion coefficients are chosen randomly in interval $[1, 2]$ and the duration of diffusion ΔT_l is chosen randomly in $[5, 20]$. The number of intermittent diffusion intervals m are specified in each example.

Example 1

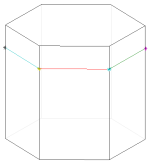
The first example computes the shortest path between two points on a hexagonal prism with side length $\sqrt{3}$ and base length 1. In one realization with $m = 10$ intermittent diffusion intervals, it finds 3 minimizers among which the global one is visited 6 times.



(A) Occurs 6 times, L=2.600



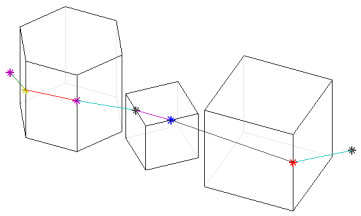
(B) Occurs 3 times, L=2.623



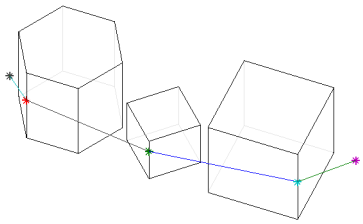
(C) Occurs once, L=3.000

Example 2

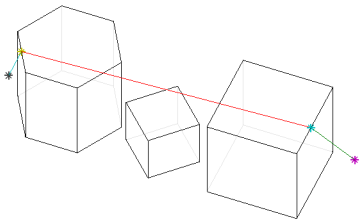
There are three obstacles in this example, two cubes and one hexagonal prism. The algorithm finds 6 local optimal paths in 20 intermittent diffusion intervals, among which the global optimal path occurs 13 times. Below we list all the local minimizers.



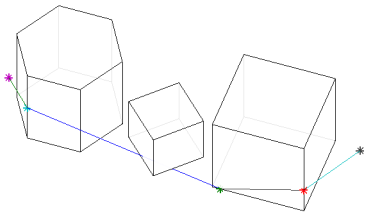
(D) Occurs 13 times, L=7.0803



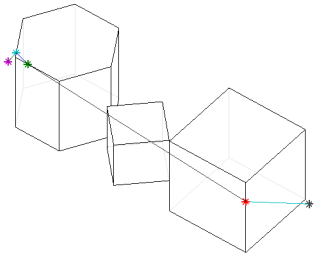
(E) Occurs 2 times, L=7.0956



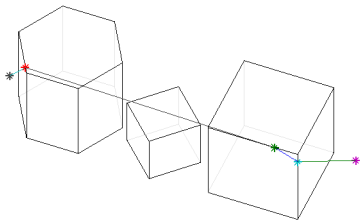
(F) Occurs 2 times, L=7.3404



(G) Occurs 1 times, L=7.3436

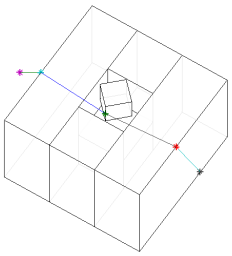


(H) Occurs 1 times, L=7.4314

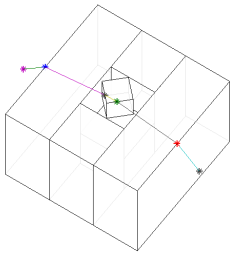


(I) Occurs 1 times, L=7.5253

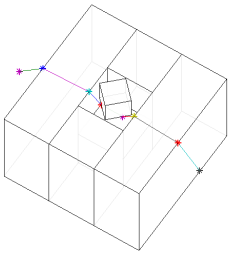
Example 3
In this example, we demonstrate that our algorithm works for non-convex obstacle without holes. One obstacle is a rotated cube and the other one is a larger cube with a indentation (not through). In 20 intermittent diffusion intervals, the algorithm finds 4 local optimal path. The global shortest path is visited 14 times.



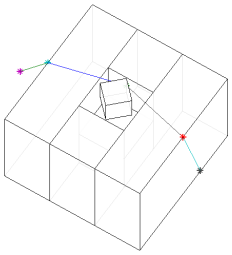
(J) Occurs 14 times, L=4.4249



(K) Occurs 2 times, L=4.4599



(L) Occurs 3 times, L=4.6176



(M) Occurs 1 times, L=4.5509

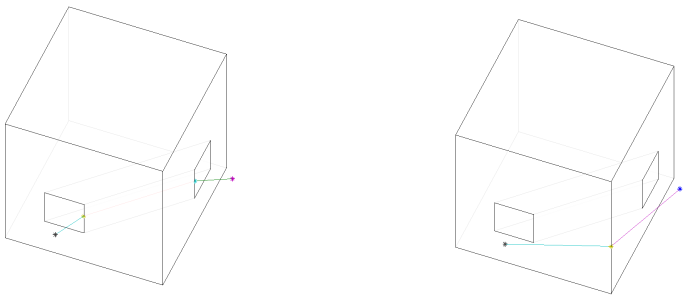
4. Polyhedron with holes

We say two paths are homotopic if one can be deformed continuously to the other while keeping its endpoints fixed. More precisely, let $\mathcal{X}$ be the space that takes away all the obstacles, i.e.

$$\mathcal{X} = \mathbf{R}^3 \setminus \bigcup P_i.$$

Two paths f_0, f_1 are path-homotopic if there exists a family of paths $f_t \colon [0,1] \to \mathcal{X}$ such that

(1) $f_t(0) = x_0$ and $f_t(1) = x_1$ are fixed.
(2) the map $F \colon [0,1] \times [0,1] \to \mathcal{X}$ given by $F(s,t) = f_t(s)$ is continuous.



(N) L=1.1543 (O) L=1.2659

FIGURE 3. Shortest path with tunneled cube

Intuitively, two paths are homotopic if one can be continuously transformed to the other without passing through the obstacles. Path-homotopic is a equivalence relation. Thus one can divide all path into equivalence classes. It is easy to see the following

Theorem 2. *If all the obstacles have Euler characteristic 2, then there is only one path-homotopic equivalence class in the set of feasible paths F.*

Proof. Since each P_i has Euler characteristic 2, P_i is homotopy to 2-dimensional sphere S^2. Notice that $R^3 - D^2$ where D^2 is the 2-dimensional disk is trivial. Therefore, any two path in $R^3 - D^2$ are path-homotopic. Same result holds for R^3 taking away n disks. □

With this result, it is simple to obtain the results in Theorem 1.

Proof of Theorem 1. Theorem 2 guarantees that our algorithm is able to visit all possible paths from any initialization. Therefore, by employing intermittent diffusion, equation (7) converges to the global minimizer with probability $1 - \epsilon$ for suitable $\sigma(t)$. □

However, on the contrary, if the obstacle contains holes, for example, a triangulated torus, there would be multiple equivalence classes. For illustration, see the following tunneled cube in Figure (3). The shortest path through the hole is 1.1543 while the one doesn't has length 1.2659. By slightly changing the position of the hole, the shortest path would be the one that not pass through it. Therefore, multiple initializations are needed to ensure that all possible equivalence classes are covered.

A simple idea we can use is to block the homotopy equivalence class the current path belongs and then reinitialize. A block can be formed ,for example, by identifying the entrance of the path followed by deleting all the vertices on it. Those vertices will not be used in the reinitialization which forces the new path to a different homotopy class. After it settles down at the global minimizer in the current homotopy class, the path is reinitialized and the algorithm is repeated to get a different global minimizer. This procedure is repeated until all homotopy equivalence classes are visited. The two paths in the above example are obtained in this method.

However, there are two problems with this approach. First of all, the block is often difficult to form because which vertices should be removed is a complicated matter, for instance, a well triangulated torus as follows. Second, the number of different homotopy classes we need to visit is unknown in prior. For example, topologically,

there are infinitely many homotopy classes for a smooth torus and the shortest path could wind the torus arbitrary times. In Figure (5), the shortest path winds the torus twice.

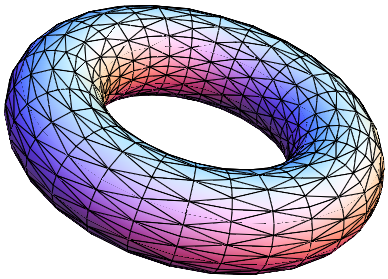


FIGURE 4. A triangulated torus

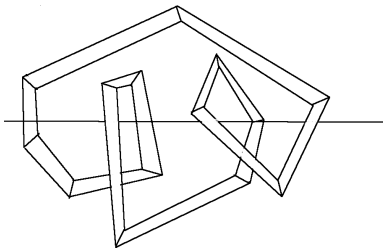


FIGURE 5. A shortest path winding a torus twice

A different approach is to use already established approximation method, for example [10] as described in the introduction section, to initialize the path. Those algorithms are able to obtain a path that has length $1 + \epsilon$ times the length of the shortest path. Here ϵ depends on the mesh size. If the mesh size is sufficiently small, the initialized path and the global minimizer will be in the same homotopy class. However, the choice of the grid size is a critical and often hard to determine.

As discussed above, our method still applies for polyhedrons with holes provided that appropriate initializations are taken. Although initialization is a complicated matter, simple ideas usually work for most cases. We conclude our discussion here and leave the improvement of initialization methods to our future work.

REFERENCES

[1] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science*, pages 49–60. IEEE, 1987.

[2] Joonsoo Choi, Jürgen Sellen, and Chee-Keng Yap. Precision-sensitive euclidean shortest path in 3-space. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 350–359. ACM, 1995.

[3] S.-N. Chow, T.-S. Yang, and H. Zhou. Global Optimizations by Intermittent Diffusion. *National Science Council Tunghai University Endowment Fund for Academic Advancement Mathematics Research Promotion Center*, page 121, 2009.

[4] Shui-Nee Chow, Jun Lu, and Haomin Zhou. Finding the shortest path by evolving junctions on obstacle boundaries (e-job): An initial value ode's approach. *Applied and computational harmonic analysis*, 2013.

[5] Laxmi P Gewali, Simeon Ntafos, and Ioannis G Tollis. Path planning in the presence of vertical obstacles. *Robotics and Automation, IEEE Transactions on*, 6(3):331–341, 1990.
[6] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.
[7] K. Jiang, LS Seneviratne, and SWE Earles. Finding the 3d shortest path with visibility graph and minimum potential energy. In *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 1, pages 679–684. IEEE, 1993.
[8] S. M. LaValle. *Planning algorithms*. Cambridge Univ Pr, 2006.
[9] J. S. B. Mitchell. Shortest path and networks. *Handbook of Discrete and computational geometry*, pages 755–778, 1997.
[10] C.H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20(5):259–263, 1985.
[11] John H Reif and James A Storer. Shortest paths in euclidean space with polyhedral obstacles. Technical report, DTIC Document, 1985.
[12] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591, 1996.
[13] A Sharir and Avikam Baltsan. On shortest paths amidst convex polyhedra. In *Proceedings of the second annual symposium on Computational geometry*, pages 193–206. ACM, 1986.
[14] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–628, 2005.

School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A., {chow, jlu39, hmzhou}@math.gatech.edu.