The Shortest Path AMID 3-D Polyhedral Obstacles

Shui-Nee Chow, Jun Lu, and 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.

Keywords Path planning · Stochastic differential equations · Intermittent diffusion · Obstacle avoidance · Global optimization

Dedicated to Professor RAYMOND H. CHAN on the occasion of his 60th birthday

This work is partially supported by NSF Awards DMS-1419027, DMS-1620345, and ONR Award N000141310408

S.-N. Chow · J. Lu · H.-M. Zhou (✉)
School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: hmzhou@math.gatech.edu

S.-N. Chow
e-mail: chow@math.gatech.edu

J. Lu
e-mail: junlu0@icloud.com

X.-C. Tai et al. (eds.), *Mathematical Methods in Image Processing and Inverse Problems*,
Springer Proceedings in Mathematics & Statistics 360,
https://doi.org/10.1007/978-981-16-2701-9_10

1 Introduction

Finding the shortest path in the presence of obstacles is one of the fundamental problems in path planning and robotics. It is one of the enabling technologies that make it possible for robots or UAVs to traverse cluttered environments. 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 (See for example [6, 12] and references therein), 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 [9] 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 [12, 14] for a survey of the results and references therein. However, Canny and Rief [2] 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 [15] 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 [3] with complexity $O(\log(\frac{1}{\epsilon}) + P(1/\delta))$ for some polynomial P. The idea is to apply the approach in [15] 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.

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 [19] 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 [7]. A different method, called unfolding technique, was introduced in [17] 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 differential equation based methods have been proposed to tackle the shortest path problem with obstacles having smooth boundaries. For example, a path evolution method finds the solution by solving a 2-point boundary value ordinary differential equation (ODE), resulting in locally 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 the fast marching method [18] or the fast sweeping method [20].

In [5], 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 address the drawback that the gradient descent method usually gets stuck at local minimizers, 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 temporally 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. More specifically, by leveraging the recent studies of convergence rate of Fokker-Planck equations [1, 4, 10, 13], it has been shown that the time complexity of E-JOB is $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$.

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 geodesic on the boundaries between two given points. In [5], 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 concatenation of line segments whose ending points are located on polyhedron edges. And to determine the geodesic is equivalent to determining 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, each junction moves in a 1-D interval(edge). 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 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 [3] in the sense that both employ a gradient descent strategy. However, before the strategy can be applied, the method in [3] 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 [15]). 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 Sect. 2, we give the derivation of the algorithm following the ideas presented in [5]. The algorithm is then presented whose details follow afterwards. In Sect. 3, we give several interesting examples. Finally, we discuss the topological issues 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

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

where θ is a scalar in $[0, 1]$. 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 length of the geodesic on P_k connecting x and y. For instance, for any x and y on the same surface of 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

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

It is worth mentioning that both θ and $\mathbf{u}_i, \mathbf{v}_i$ are dynamic as we optimize over x_is.

2.2 Structure of the Shortest Path

A path is a curve $\gamma \in \mathbf{R}^3$, which is a continuous map

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

We denote $L(\gamma)$ the Euclidean length of the path γ. We are concerned with the set of feasible paths $\mathbf{F}$, i.e. paths that do not 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 [5], 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

$$\mathcal{L}(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} (J(x_i) + \|x_1 - x_0\| + \|x_{n+1} - x_n\|). \tag{3}$$

Again all x_is are on the edges of the obstacles. 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 globally optimal path, we adopt a strategy called Intermittent Diffusion, i.e. we evolve the following SDE

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

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 $[S_l, T_l]$. We note that adding $dW(t)$, the so-called white noise, to the SDE corresponds to having a diffusion process in the classical stochastic theory. Thus adding white noise perturbations to the SDE on discontinuous intervals $[S_l, T_l]$ is like adding diffusion process intermittently. For convenience, we call $[S_l, T_l$ an intermittent diffusion interval. In this paper, the intervals S_l, T_l, σ_l are chosen to be

random and the magnitude, $\{\sigma_l\}$, of the random perturbations are selected according to the following theorem. For more details, see [4].

Theorem 1 *If all the obstacles have Euler characteristic 2, then for any small number ϵ, there exists $\tau > 0, \sigma_0 > 0$ and integer $m_0 > 0$ such that if $T_i - S_i > \tau$, $\sigma_i < \sigma_0$ and $m > m_0$, then Eq. (7) converges to a global minimizer with probability $1 - \epsilon$.*

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 Eq. (7).

2.4.1 Discretization of SDE

We use the forward Euler method to discretize equation (7)

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

where $\xi \sim N(0, 1)$ is a normal random variable. Notice the θ_is are updated alternatingly in the Gauss-Seidel fashion.

2.4.2 Initialization

We can use the optimal path whose junctions are restricted to 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) [11]. 3DRVG consists of connected planes as opposed to straight line segments which becomes complicated when there are more than one obstacles. 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's 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 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 connect 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. As an example, see the following illustration (Fig. 1) 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 $(x_1, x_2, u = x, x_4)$. 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).

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 $\Delta\theta$.

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, i.e. on the same face as 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 (Fig. 2). The feasible directions 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).

Case 3. $x = (\mathbf{u}, \mathbf{v}, \theta = 1)$ is an exterior junction, and two of its neighbors, z and y, are on other obstacles. There are two scenarios. One is when x approaching $\mathbf{u}$, we remove x from the junction list and add new points on the edges adjacent to $\mathbf{u}$ except $\mathbf{uv}$. This is the same scenario as that illustrated in Case 1. The other is that one can directly connect z and y, and this involves adding and removing junctions as be discussed in the following subsection.

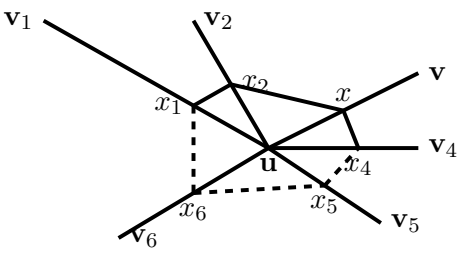


Fig. 1 Movement of interior junction

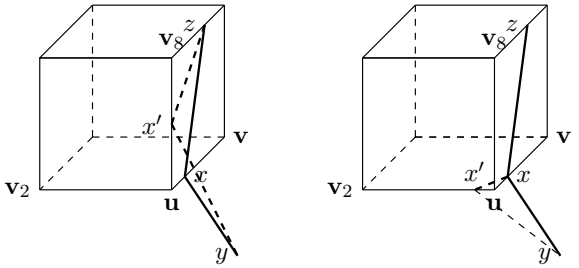


Fig. 2 Movement of exterior junction x. The left figure corresponds to case (i) and the right corresponds to case (ii)

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 $\overline{xy}$ intersects with obstacle $P_{k_1}, P_{k_2}, \cdots, P_{k_r}$ after evolution, 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 shortens 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. The determination of whether a line segment $\overline{xy}$ intersects with a face can be done by checking whether the intersection point of the line containing $\overline{xy}$ and the surface containing the face lies on the face or not.

2.5 Algorithm

We present our algorithm below

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

1 Initialization. Find the initial set U of junction points.
2 **for** $l = 1 : m$
3 $U_l = U$;
4 **for** $x_i = (\mathbf{u}, \mathbf{v}, \theta_i^0) \in U_l$
5 **for** $j = 1 : \Delta T_l$
6 Update x according to (7), i.e. $\theta_i^{j+1} = \theta_i^j + (-\nabla J(\theta_i^j) + \sigma_l \sqrt{\Delta t}\xi)\Delta t$;
7 Update set U_l, i.e. remove junctions from or add junctions to U_l;
8 **end**
9 **while** $|\theta_{i+1}^{j+1} - \theta_i^j| > \epsilon$ (or other convergence criterion)
10 Update x according to (6), i.e. $\theta_i^{j+1} = \theta_i^j - \nabla J(\theta_i^j)\Delta t$;
11 Update set U_l;
12 **end**
13 **end**
14 **end**
15 Compare U_ls and set $U_{opt} = \text{argmin}_{l \leq m} \mathcal{L}(U_l)$.

2.6 Complexity Analysis

We now give a brief analysis of the algorithm. Following [16], instead of discussing the algebraic complexity of the algorithm, we will consider the running time in order to achieve certain relative error ϵ.

(1) The initialization is done by constructing the visibility graph and Dijkstra's algorithm. Constructing the visibility graph takes $O(|V|^2)$ while Dijkstra's algorithm takes $O(|E| + |V| \log |V|)$. These two steps are exact in the sense that the complexities do not depend on ϵ. Therefore, they are not counted in the final complexity.
(2) Inner loop line 5–8 takes $O(\Delta T_l)$ time. This is because Eq. (7) takes constant time, and so does adding or removing junctions.
(3) Inner loop line 9-12 takes $T(\epsilon)$ time where $T(\epsilon)$ denotes the number of iterations required until the error is less than ϵ. If we assume the Hessian matrix of the gradient is nondegenerate, which is the case for all polyhedral obstacles [3], then $T(\epsilon) = O(\log \frac{1}{\epsilon})$.

Table 1 Complexity comparison to other Algorithms

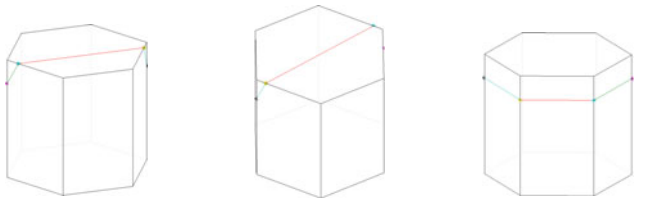
Algorithm	Complexity
A^*	$O((\frac{1}{\epsilon})^3 \log \frac{1}{\epsilon})$
Papadimitriou [16]	$O(\frac{1}{\epsilon})$
Choi et. al. [3] (When the shortest path is not unique.)	$O(\frac{1}{\epsilon})$
Choi et. al. [3] (When the shortest path is unique.)	$O(\log \frac{1}{\epsilon})$

Let $\Delta T = \max_{i \le l} \Delta T_i$. Then the total running time is $O(m(\Delta T + \log \frac{1}{\epsilon}))$. From [4], it can be shown that in order to obtain the desired successful probability $1 - \delta$, the number of realizations must be of order $O(\log \frac{1}{\delta})$. Therefore, the complexity is $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$. Table 1 shows a complexity comparison with some existing methods.

3 Numerical Examples

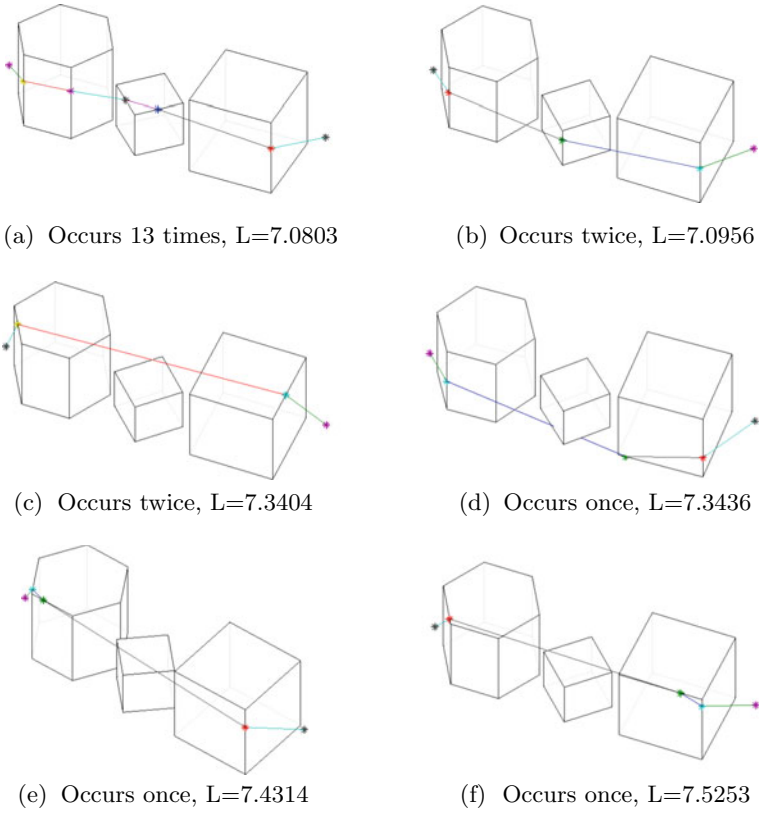
We show several examples in this section to illustrate the paths obtained by our algorithm. The diffusion coefficients are chosen randomly in interval [1, 2] and the duration of diffusion ΔT_l is chosen randomly in [5, 20]. The parameter m, the number of intermittent diffusion intervals $\{[S_l, T_l]\}_1^m$, on which the random perturbations are added to the process, 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, as illustrated in the left plot in Fig. 3, is visited 6 times. In this example, one can easily enumerate all possible combinations to conclude the path obtained with length $L = 2.6$ is the global optimal solution.



(a) Occurs 6 times, L=2.600 (b) Occurs 3 times, L=2.623 (c) Occurs once, L=3.000

Fig. 3 Example 1



(a) Occurs 13 times, L=7.0803 (b) Occurs twice, L=7.0956

(c) Occurs twice, L=7.3404 (d) Occurs once, L=7.3436

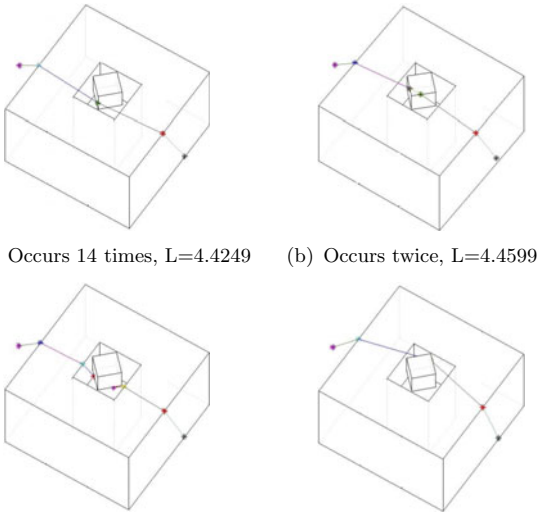
(e) Occurs once, L=7.4314 (f) Occurs once, L=7.5253

Fig. 4 Example 2

Example 2 There are three obstacles in this example (Fig. 4), 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.

Example 3 In this example (Fig. 5), we demonstrate that our algorithm works for non-convex obstacles without holes. One obstacle is a rotated cube and the other one is a larger cube with an unpenetrated indentation. In 20 intermittent diffusion intervals, the algorithm finds 4 locally optimal path. The globally shortest path is visited 14 times.

Fig. 5 Example 3



(a) Occurs 14 times, L=4.4249 (b) Occurs twice, L=4.4599

(c) Occurs 3 times, L=4.6176 (d) Occurs once, 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 $f_t : [0, 1] \to \mathcal{X}$ such that

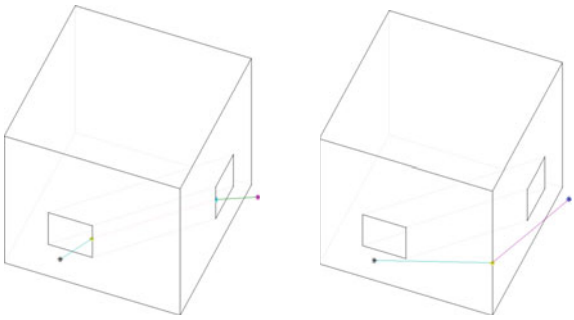
(1) $f_t(0) = x_0$ and $f_t(1) = x_1$ are fixed.
(2) the map $F : [0, 1] \times [0, 1] \to \mathcal{X}$ given by $F(s, t) = f_t(s)$ is continuous.

Intuitively, two paths are homotopic if one can be continuously transformed to the other without passing through the obstacles. Path-homotopy is an equivalence relation. Thus one can divide all paths 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-homotopy equivalence class in the set of feasible paths F.*

Proof Since each P_i has Euler characteristic 2, P_i is homotopic to 2-dimensional sphere S^2. Notice that $R^3 - B^3$ where B^3 is the 3-dimensional ball is simple-connected. Therefore, any two path in $R^3 - B^3$ are homotopic [8]. Same result holds for R^3 taking away n balls.

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



(a) L=1.1543 (b) L=1.2659

Fig. 6 Shortest path with tunneled cube

Proof of Theorem 1 Theorem 2 guarantees that our algorithm is able to visit all possible paths from any initialization. The rest of the theorem is simply the statement from [4] and hence omitted.

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. The shortest path through the hole is 1.1543 while the one that doesn't penetrate the hole has length 1.2659. By slightly changing the position of the hole, the shortest path would be the one that does 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 to and then reinitialize. "Blocking" means deleting some vertices of the obstacles such that the reinitialization will force the new path to a different homotopy class. After the gradient descent 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 by this method (Fig. 6).

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 (Fig. 7). Second, the number of different homotopy classes we need to visit is unknown in advance. For example, topologically, there are infinitely many homotopy classes for a smooth torus and the shortest path could wind the torus arbitrary times. In Fig. 8, the shortest path winds the torus twice.

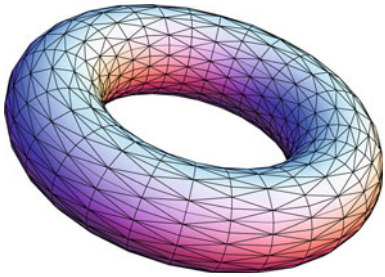


Fig. 7 A triangulated torus

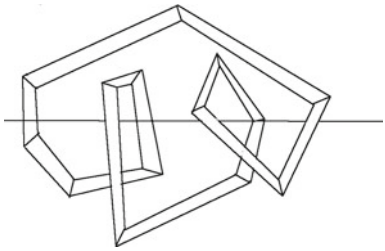


Fig. 8 A shortest path winding a torus twice

A different approach is to use an already established approximation method, for example [15] 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.

5 Future Work

The method we propose in this work can be equally applied to a general class of problems. In detail, consider the following problem

$$\gamma_{opt} = \text{argmin}_{\gamma\in\mathbf{F}}\, L(\gamma). \tag{9}$$

Here L is a general functional on $\mathbf{F}$ with the form

$$L(\gamma) = \int_0^1 l(|\dot{\gamma}(\theta)|)\,\mathrm{d}\theta \tag{10}$$

where $l(x)$ is a convex function. The Euclidean length of the path, which we consider in this paper, simple corresponds to the case where $l(x) = x$. It turns out that in this general setting, the optimal path has the same simple structure as mentioned in this paper. Therefore, the method can be applied without any essential modification. An example is $l(x) = x^2$, in which case the functional L represents the oil consumption of a car. We leave the direction for future exploration.

References

1. A. Arnold, P. Markowich, G. Toscani, A. Unterreiter, On convex sobolev inequalities and the rate of convergence to equilibrium for fokker-planck type equations (2001)
2. J. Canny, J. Reif, New lower bound techniques for robot motion planning problems, in *28th Annual Symposium on Foundations of Computer Science*, pp. 49–60. IEEE (1987)
3. J. Choi, J. Sellen, C.-K. Yap, Precision-sensitive euclidean shortest path in 3-space, in *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pp. 350–359. ACM (1995)
4. S.-N. Chow, T.-S. Yang, H. Zhou, Global optimizations by intermittent diffusion. *National Science Council Tunghai University Endowment Fund for Academic Advancement Mathematics Research Promotion Center*, p. 121 (2009)
5. S.-N. Chow, J. Lu, H.M. Zhou, Finding the shortest path by evolving junctions on obstacle boundaries (E-JOB): An initial value ODE's approach. J. Appl Comput Harmon Anal. **35**(1), 156–176 (2013)
6. F. Fahimi, *Autonomous Robots: Modeling, Path Planning, and Control* (Springer, 2008)
7. L.P. Gewali, S. Ntafos, I.G. Tollis, Path planning in the presence of vertical obstacles. IEEE Trans. Robot. Autom. **6**(3), 331–341 (1990)
8. A. Hatcher, *Algebraic Topology* (2002)
9. J. Hershberger, S. Suri, An optimal algorithm for Euclidean shortest paths in the plane. SIAM J. Comput. **28**(6), 2215–2256 (1999)
10. R. Holley, D. Stroock, Logarithmic sobolev inequalities and stochastic ising models. J. Statist. Phys. **46**(5), 1159–1194 (1987)
11. K. Jiang, L.S. Seneviratne, S.W.E. 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*, vol. 1, pp. 679–684. IEEE (1993)
12. S. M. LaValle. *Planning Algorithms* (Cambridge University Press, 2006)
13. P.A. Markowich, C. Villani, On the trend to equilibrium for the fokker-planck quation: an interplay between physics and functional analysis. Mat. Contemp. **19**, 1–29 (2000)
14. J.S.B. Mitchell, Shortest path and networks. *Handbook of Discrete and Computational Geometry*, pp. 755–778 (1997)
15. C.H. Papadimitriou, An algorithm for shortest-path motion in three dimensions. Inf. Process. Lett. **20**(5), 259–263 (1985)
16. C.H. Papadimitriou, An algorithm for shortest-path motion in three dimensions. Inf. Process. Lett. **20**(5), 259–263 (1985)
17. J.H. Reif, J.A. Storer, Shortest paths in euclidean space with polyhedral obstacles. Technical report, DTIC Document (1985)
18. J.A. Sethian, A fast marching level set method for monotonically advancing fronts. Proc. Natl Acad. Sci. **93**(4), 1591 (1996)
19. A. Sharir, A. Baltsan, On shortest paths amidst convex polyhedra, in *Proceedings of the Second Annual Symposium on Computational Geometry*, pp. 193–206. ACM (1986)
20. H. Zhao, A fast sweeping method for eikonal equations. Math. Computat. **74**(250), 603–628 (2005)