Approximate Shortest Path Calculations in Simple Polyhedra

Fajie Li[1] and Reinhard Klette[2]

[1] College of Computer Science and Technology
Huaqiao University, Quanzhou, Fujian, China
[2] Computer Science Department, The University of Auckland
Private Bag 92019, Auckland 1142, New Zealand

Abstract. This paper considers the calculation of a Euclidean shortest path (ESP) in a three-dimensional (3D) polyhedral space Π. We propose an approximate $\kappa(\varepsilon) \cdot \mathcal{O}(M|V|)$ 3D ESP algorithm (excluding preprocessing), with preprocessing time complexity $\mathcal{O}(M|E| + |S| + |V| \log |V|)$, for solving a special, but 'fairly general' case of the 3D ESP problem, where Π does not need to be convex, V and E are the sets of vertices and edges of Π, respectively, and S is the set of faces (triangles) of Π; M is the maximal number of vertices of a so-called critical polygon; $\kappa(\varepsilon) = (L_0 - L)/\varepsilon$ where L_0 is the length of an initial path and L is the true (i.e., optimum) path length. The given algorithm solves approximately three (previously known to be) NP-complete or NP-hard 3D ESP problems in time $\kappa(\varepsilon) \cdot \mathcal{O}(k)$, where k is the number of layers in a stack, which is introduced in this paper as being the *problem environment*. The proposed approximation method has straightforward applications for ESP problems when analyzing polyhedral objects (e.g., in 3D imaging), of for 'flying' over a polyhedral terrain.

1 Introduction and Related Work

There exist already several approximation algorithms for 3D ESP calculations, and we briefly recall those. Pioneering the field, [12] presents an

$$\mathcal{O}(n^4(L + \log(n/\varepsilon))^2/\varepsilon^2)$$

algorithm for the general 3D ESP problem, where n is the number of polyhedral scene elements (that is, vertices, edges, and faces of the polyhedron), ε the accuracy of the algorithm, and L the number of bits of the largest integer describing the coordinates of the polyhedral scene elements. This was followed by [6], which presents an approximation algorithm for computing an $(1 + \varepsilon)$-shortest path from p to q in time

$$\mathcal{O}(n^2\lambda(n)\log(n/\varepsilon)/\varepsilon^4 + n^2 \ \log \ nr \ \log(n \ \log r))$$

where r is the ratio of the Euclidean distance $d_e(p, q)$ to the length of the longest edge of any given obstacle, and

$$\lambda(n) = \alpha(n)^{\mathcal{O}(\alpha(n)^{\mathcal{O}(1)})}$$

where $\alpha(n) = A^{-1}(n, n)$ is the inverse Ackermann function [9], which grows very slowly (because A grows very rapidly). Let there be a finite set of polyhedral obstacles in $\mathbb{R}^3$. Let p, q be two points outside of the union of all obstacles. Assume that $0 < \varepsilon < 1$; [7] gives an $\mathcal{O}(\log(n/\varepsilon))$ algorithm to compute an $(1+\varepsilon)$-shortest path from p to q such that it avoids the interior of any obstacle. The algorithm is based on a subdivision of $\mathbb{R}^3$ which is computed in $\mathcal{O}(n^4/\varepsilon^6)$. Given a convex partition of the free space, [2] presents an $\mathcal{O}((n/\varepsilon^3)(\log 1/\varepsilon)(\log n))$ algorithm for the general 3D ESP problem. Altogether, the task of finding efficient and easy to implement solutions in this field is certainly challenging; see, for example, [10] saying on page 666 the following: "*The problem is difficult even in the most basic Euclidean shortest-path problem (ESP) in a three-dimensional polyhedral domain Π, and even if the obstacles are convex, or the domain Π is simply connected.*" In this paper, we apply a simplified version of a rubberband algorithm (see [8] for a general introduction to those algorithms) to present an approximate

$$\kappa(\varepsilon) \cdot \mathcal{O}(M|V|) + \mathcal{O}(M|E| + |S| + |V| \log |V|)$$

algorithm for ESP calculation when Π is a (type 2, see Definition 2 below) simply connected polyhedron which is not necessarily convex. V and E are the sets of vertices and edges of Π, respectively; S is the set of faces (triangles) of Π; M is the maximal number of vertices of the critical polygon (see Definition 1 below); $\kappa(\varepsilon) = (L_0 - L)/\varepsilon$, where L_0 is the length of an initial path and L the true (i.e., optimum) path length.

The given algorithm solves approximately three NP-complete or NP-hard 3D ESP problems in time $\kappa(\varepsilon) \cdot \mathcal{O}(k)$, where k is the number of layers in a stack, which is introduced as a 'problem environment' below. Our algorithm has straightforward applications for ESP problems when analyzing polyhedral objects (e.g., in 3D imaging; for the extensive work using geodesics we just cite [16] as one example), or for 'flying' over a polyhedral terrain. The best known result for the latter problem is due to [17] by proposing an $\mathcal{O}((n/\varepsilon)(\log n)(\log \log n))$ algorithm for computing a $(2^{(p-1)/p} + \varepsilon)$-approximation of an L_p-shortest path above a polyhedral terrain.

Section 2 provides necessary definitions and theorems. Section 3 describes our algorithm. Section 4 gives the time complexity of the algorithm. Section 5 illustrates the algorithm by some examples. Section 6 concludes the paper.

2 Basics

We denote by Π a *simple polyhedron* (i.e., a compact polyhedral region which is homeomorphic to a unit ball) in the 3D Euclidean space, which is equipped with an xyz Cartesian coordinate system. Let E be the set of edges of Π; $V = \{v_1, v_2, \ldots, v_n\}$ the set of vertices of Π. For $p \in \Pi$, let π_p be the plane which is incident with p and parallel to the xy-plane. The intersection $\pi_p \cap \Pi$ is a finite set of simple polygons; a singleton (i.e., a set only containing a single point) is considered to be a degenerate polygon. – Now let P be such a simple polygon, defined by p and Π.

Definition 1. *Any simple polygon P, being one connected component of $\pi_p \cap \Pi$, is a* critical polygon *of Π (with respect to p).*

Any vertex p defines in general a finite set of critical polygons. The notion of a critical polygon is also generalized as follows: We assume a simply connected (possibly unbounded) polyhedron Π, and we allow that the resulting (generalized) critical polygons may also be unbounded. For example, a generalized critical polygon may have a vertex at infinity, or it can be the complement of a critical polygon, as specified in Definition 1. (Section 5 will also make use of generalized critical polygons.)

Definition 2. *We say that a simple polyhedron Π is a* type 1 *polyhedron iff any vertex p defines exactly one convex critical polygon. We say that a simple polyhedron Π is a* type 2 *polyhedron iff any vertex p defines exactly one simple critical polygon.*

Obviously, each type 1 simple polyhedron is also a type 2 simple polyhedron. Our main algorithm below applies to type 2 simple polyhedra. – We recall some concepts introduced in [11]. Let (x_0, y_0, z_0) be a point in 3D space. Let

$$S_1 = \{(x, y, z_0) : x_0 \leq x < \infty \wedge y_0 \leq y < \infty\}$$
$$S_2 = \{(x, y, z_0) : -\infty < x \leq x_0 \wedge y_0 \leq y < \infty\}$$
$$S_3 = \{(x, y, z_0) : -\infty < x \leq x_0 \wedge -\infty < y \leq y_0\}$$
$$S_3 = \{(x, y, z_0) : x_0 \leq x < \infty \wedge -\infty < y \leq y_0\}$$

S_i is called a *q-rectangle of type i*, where i = 1, 2, 3, 4. Furthermore, let (x_1, y_1, z_0) be a point in 3D space such that $x_1 > x_0$ and $y_1 > y_0$. Let

$$S_h = \{(x, y, z_0) : -\infty < x < \infty \wedge y_0 \leq y \leq y_1\}$$
$$S_v = \{(x, y, z_0) : x_0 \leq x \leq x_1 \wedge -\infty < y < \infty\}$$

Finally, let

$$S_{h_1} = \{(x, y, z_0) : x_0 \leq x < \infty \wedge y_0 \leq y \leq y_1\}$$
$$S_{h_2} = \{(x, y, z_0) : -\infty < x \leq x_0 \wedge y_0 \leq y \leq y_1\}$$
$$S_{v_1} = \{(x, y, z_0) : x_0 \leq x \leq x_1 \wedge y_0 \leq y < \infty\}$$
$$S_{v_2} = \{(x, y, z_0) : x_0 \leq x \leq x_1 \wedge -\infty < y \leq y_0\}$$

S_h, S_v, S_{h_j}, and S_{v_j} are called *horizontal or vertical strips*, for j = 1, 2. According to their geometric shape, we notice that

(i) S_1 [S_2, S_3, S_4] is unbounded in direction $(+x, +y)$ [$(-x, +y)$, $(-x, -y)$, $(+x, -y)$];
(ii) S_h [S_v] is unbounded in direction $\pm x$ [$\pm y$];
(iii) S_{h_1} [S_{h_2}, S_{v_1}, S_{v_2}] is unbounded in direction $+x$ [$-x$, $+y$, $-y$].

S_i, S_h, S_v, S_{h_j}, and S_{v_j} are also called *axis-aligned rectangles*, where $i = 1, 2, 3, 4$ and $j = 1, 2$. The stack $\mathcal{S}$ of axis-aligned rectangles is called *terrain-like* if, for at least one of the four directions $-x$, $+x$, $-y$, or $+y$, each rectangle in $\mathcal{S}$ is unbounded.

Let s_0, s_1 and s_2 be three segments. Let $p_i = (p_{i_1}, p_{i_2}, p_{i_3})$ be on s_i, for $i = 0, 1, 2$. Then we have the following (which we could not find elsewhere; so we state and prove it here for completeness):

Lemma 1. *A unique point p_1', which satisfies*

$$d_e(p_1', p_0) + d_e(p_1', p_2) = \min\{p_1 : d_e(p_1, p_0) + d_e(p_1, p_2) \wedge p_1 \in s_1\}$$

can be computed in $\mathcal{O}(1)$ time.

Proof. Let the two endpoints of s_1 be $a_1 = (a_{1_1}, a_{1_2}, a_{1_3})$ and $b_1 = (b_{1_1}, b_{1_2}, b_{1_3})$. Let $p_0 = (p_{0_1}, p_{0_2}, p_{0_3})$. Point p_1 can be written as $(a_{1_1} + (b_{1_1} - a_{1_1})t, a_{1_2} + (b_{1_2} - a_{1_2})t, a_{1_3} + (b_{1_3} - a_{1_3})t)$. The formula

$$d_e(p_1, p_0) = \sqrt{\sum_{i=1}^{3} [(a_{1_i} - p_{0_i}) + (b_{1_i} - a_{1_i})t]^2}$$

can be simplified: Without loss of generality, we can assume that s_1 is parallel to one of the three coordinate axes. It follows that only one element of the set $\{b_{1_i} - a_{1_i} : i = 1, 2, 3\}$ is not equal to 0, and the other two are equal to 0. Thus, we can assume that

$$d_e(p_1, p_0) = |A_1|\sqrt{(t + B_1)^2 + C_1}$$

where A_1, B_1 and C_1 are functions of a_{1_i}, b_{1_i} and p_{0_i}, for i = 0, 1, 2. – Analogously,

$$d_e(p_1, p_2) = |A_1|\sqrt{(t + B_2)^2 + C_2}$$

where B_2 and C_2 are functions of a_{1_i}, b_{1_i} and p_{2_i}, for i = 0, 1, 2. In order to find an optimum point $p_1' \in e_1$ such that

$$d_e(p_1', p_0) + d_e(p_1', p_2) = \min\{p_1 : d_e(p_1, p_0) + d_e(p_1, p_2), p_1 \in s_1\}$$

we can solve the equation

$$\frac{\partial(d_e(p_1, p_0) + d_e(p_1, p_2))}{\partial t} = 0$$

The unique solution is $t = -(B_1C_2 + B_2C_1)/(C_2 + C_1)$. □

In what follows, Π is a type 2 simple polyhedron. For a simple polygon P, let P° be its topological interior, $P^\bullet$ the closure of P°, and $\partial P = P^\bullet \setminus P^\circ$ (the frontier of $P^\bullet$). Let $\rho(p, q)$ be a path from p to q, and $p_x(p_y, p_z)$ the $x(y, z)$-coordinate of p.

Let q_1, $q_2 \in \mathbb{R}^3$ such that $q_1 \neq q_2$. Let the coordinates of q_i be equal to (x_i, y_i, z_i), where i = 1, 2. We use (standard) lexicographic order: q_1 *is before* q_2 iff $q_{1x} < q_{2x}$, or $q_{1x} = q_{2x}$ and $q_{1y} < q_{2y}$, or $q_{1x} = q_{2x}$ and $q_{1y} = q_{2y}$ and $q_{1z} < q_{2z}$. If q_1 is before q_2, then we write that $\min\{q_1, q_2\} = q_1$, otherwise $\min\{q_1, q_2\} = q_2$. Let s_1 and s_2 be two segments. Let p_1 and p_2 be two points such that $p_i \notin s_j$, where i, j = 1, 2. Let S = $\{p : p \in s_1 \vee p \in s_2\}$. Then we have the following

Lemma 2. *A point $p \in S$ such that*

$$d_e(p_1, p) + d_e(p_2, p) = \min\{d_e(p_1, q) + d_e(p_2, q) : q \in S\}$$

can be computed in $\mathcal{O}(1)$ time.

Proof. By Lemma 1, there is a unique point $q_i \in s_i$ such that

$$d_e(p_1, q_i) + d_e(p_2, q_i) = \min\{d_e(p_1, q) + d_e(p_2, q) : q \in s_i\}$$

where i = 1, 2. Let s = q_1q_2. We can find the required point p as follows:
Case 1. $d_e(p_1, q_1) + d_e(p_2, q_1) < d_e(p_1, q_2) + d_e(p_2, q_2)$. Let p = q_1.
Case 2. $d_e(p_1, q_1) + d_e(p_2, q_1) > d_e(p_1, q_2) + d_e(p_2, q_2)$. Let p = q_2.
Case 3. $d_e(p_1, q_1) + d_e(p_2, q_1) = d_e(p_1, q_2) + d_e(p_2, q_2)$. Let p = $\min\{q_1, q_2\}$. □

From the proof it also follows that such a point p is actually uniquely specified if it is obtained by comparing it with the other candidate points not only by length but also by coordinates.

Let $\{s_1, s_2, \ldots, s_m\}$ be a set of segments. Let p_1 and p_2 be two points such that $p_i \notin s_j$, where i = 1, 2, and j = 1, 2, . . ., m. Let S = $\{p : p \in s_j \wedge j = 1, 2, \ldots, m\}$. By Lemma 2, we have immediately the following

Theorem 1. *A point $p \in S$ which satisfies*

$$d_e(p_1, p) + d_e(p_2, p) = \min\{d_e(p_1, q) + d_e(p_2, q) : q \in S\}$$

can be computed in $\mathcal{O}(m)$ time.

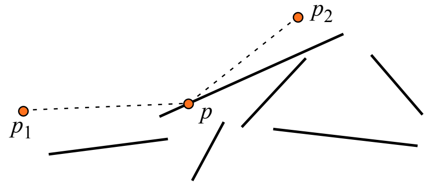

Fig. 1. Two points p_1 and p_2 and m segments.

Note that the specified point p in Theorem 1 is not unique in general. See Figure 1; for example, there could be a 'symmetric minimum'. However, using the described alphabetic order, we can select a unique minimum with respect to this order. This uniqueness is very important for the proof of the correctness of Algorithm 2 below. (Obviously, Theorem 1 is still correct if s_j is a straight line or a ray, for all or some j = 1, 2, . . ., m.)

3 ESP Calculation

We start presenting a procedure used by a rubberband algorithm (Algorithm 1 below) which is then frequently called in the main algorithm (Algorithm 2) of this section.

Let $S = \{\triangle_1, \triangle_2, \ldots, \triangle_m\}$ be the set of all faces of Π. – The following very basic Procedure 1 simply 'walks around' the polyhedron by tracing an intersection with a given plane; it is given here for completeness.

Procedure 1 (compute a sequence of vertices of the critical polygon; see Fig. 2)
Input: *A set S and a vertex $v \in V$ such that $|\pi_v \cap \Pi| > 1$.*
Output: *A sequence V_v of all vertices of the critical polygon P_v.*

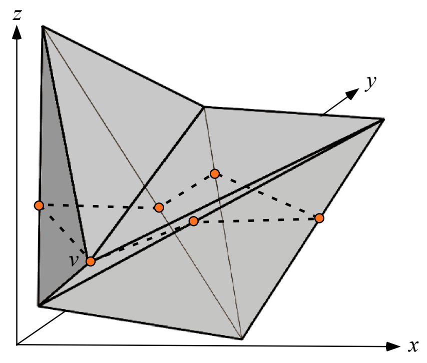

Fig. 2. The labeled vertex v identifies a sequence of six vertices of the critical polygon P_v, defined by the intersection of plane π_v with the shown Schönhardt polyhedron.

```
1: Let $S_v = \{\triangle : \triangle \in S \wedge e = uw \in E(\triangle) \wedge (u_z \leq v_z \leq w_z \vee w_z \leq v_z \leq u_z)\}$,
   and $E(S_v) = \{e : \exists \triangle \in S_v \wedge e \in E(\triangle)\}$.
2: Let $V_v = \emptyset$, and $v_1 = v$.
3: Let $V_v = V_v \cup \{v_1\}$.
4: Find a face $\triangle_1 \in S_v$ such that $v_1 \in V(\triangle_1)$, and such that there exists an edge
   $e \in E(\triangle_1)$ with $v_1 \notin e$ and $\pi_{v_1} \cap e \neq \emptyset$, and $\neq v_1$.
5: Do the following substeps:
6: **if** $|\pi_{v_1} \cap e| = 1$ (note: $e$ is not parallel to the plane $\pi_{v_1}$) **then**
7:     Let $\pi_{v_1} \cap e = v_2$, and $V_v = V_v \cup \{v_2\}$.
8: **else**
9:     Let $e = w_1w_2$ (that is, $w_1$ and $w_2$ are endpoints of edge $e$).
10:    **if** $vw_i \in E(S_v)$, for $i$ = 1 and 2 (note: the critical polygon $P_{v_1}$ is a triangle) **then**
11:        Let $V_v = V_v \cup \{w_1, w_2\}$.
12:    **else**
13:        **if** $v_1w_1 \in E(S_v)$ and $v_1w_2 \notin E(S_v)$ **then**
14:            Find a face $\triangle_2 \in S_{v_1}$ such that $w_2 \in V(\triangle_2)$, $w_1w_2 \notin E(\triangle_2)$ and there
               exists an edge $e_2 \in E(\triangle_2)$ such that $w_2 \notin e_2$ and $\pi_{w_2} \cap e_2 \neq \emptyset$.
15:            Let $\triangle_1 = \triangle_2$, $v_1 = w_2$, $e = e_2$, $V_v = V_v \cup \{w_2\}$, and go to Step 5.
16:        **else**
17:            Find a face $\triangle_2 \in S_{v_1}$ such that $w_1 \in V(\triangle_2)$, $w_1w_2 \notin E(\triangle_2)$ and there
               exists an edge $e_2 \in E(\triangle_2)$ such that $w_1 \notin e_2$ and $\pi_{w_1} \cap e_2 \neq \emptyset$.
18:            Let $\triangle_1 = \triangle_2$, $v_1 = w_1$, $e = e_2$, $V_v = V_v \cup \{w_1\}$, and go to Step 5.
19:        **end if**
20:    **end if**
21: **end if**
22: Let $\triangle_2 \in S_v$ be that face which shares edge $e$ with $\triangle_1$.
23: **if** $v_2 \neq v$ **then**
24:     Let $\triangle_1 = \triangle_2$ and $v_1 = v_2$, and go to Step 3.
25: **else**
26:     Output $V_v$, and Stop.
27: **end if**
```

The main ideas of the following rubberband algorithm (Algorithm 1) are as follows: For a start, we randomly take a point in the closure of each critical polygon to identify an initial path from p to q. Then we enter a loop; in each iteration, we optimize point p_1, then p_2, …, and finally p_k. At the end of each iteration, we check the difference between the length of the current path to that of the previous one; if it is less than ε then we stop. Otherwise, we go into the next iteration.

Algorithm 1 (a rubberband algorithm)
Input: *Two points* p *and* q*; set* $\{P^\bullet_{v_1}, P^\bullet_{v_2}, \ldots, P^\bullet_{v_k}\}$*, where* P_{v_i} *is a critical polygon of* Π*;* k *vertices* $v_i \in \partial P_{v_i}$ *such that* $p_z < v_{1_z} < \cdots < v_{k_z} < q_z$*,* i *= 1, 2, …,* k*, and there is no any other critical polygon of* Π *between* p *and* q*.*
Output: *The set of all vertices of an approximate shortest path which starts at* p*, then visits (final optimal points)* p_1*,* p_2*, …,* p_k *in that order, and finally* q*.*

```
1: Let $\varepsilon = 10^{-10}$ (the chosen accuracy).
2: for each $i \in \{1, 2, \ldots, k\}$ do
3:     Randomly take a point $p_i$ in $P^\bullet_{v_i}$.
4: end for
5: Compute the length $L_0$ of the path $\rho = (p, p_1, p_2, \ldots, p_k, q)$.
6: Let $q_1 = p$ and $i = 1$.
7: while $i < k - 1$ do
8:     Let $q_3 = p_{i+1}$.
9:     Compute [3] a point $q_2 \in P^\bullet_{p_i}$ such that
           $$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q) + d_e(q_3, q) : q \in P^\bullet_{p_i}\}$$
10:    Update $\rho$ by replacing $p_i$ by $q_2$.
11:    Let $q_1 = p_i$ and $i = i + 1$.
12: end while
13: Let $q_3 = q$.
14: Compute $q_2 \in P^\bullet_{p_k}$ such that
           $$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q) + d_e(q_3, q) : q \in P^\bullet_{p_k}\}$$
15: Update $\rho$ by replacing $p_k$ by $q_2$. [Note: Steps 13–15 are analogous to Steps 8–10.]
16: Compute the length $L$ of the updated path $\rho = (p, p_1, p_2, \ldots, p_k, q)$.
17: Let $\delta = L_0 - L$.
18: if $\delta > \varepsilon$ then
19:     Let $L_0 = L$ and go to Step 6.
20: else
21:     Stop.
22: end if
```

The set $\{P^\bullet_{v_1}, P^\bullet_{v_2}, \ldots, P^\bullet_{v_k}\}$ in Algorithm 1 is called the *step set* of the rubberband algorithm. (Identifying a correct step set is normally the main issue when defining a rubberband algorithm; for example, see [8].)

[3] If $q_1q_3 \cap P^\circ_{p_i} \neq \emptyset$, then let $q_2 = q_1q_3 \cap P^\circ_{p_i}$.

The main ideas of the following Algorithm 2 are as follows: We apply Procedure 1 to compute the step set of a rubberband algorithm as given in Algorithm 1. Then we simply apply this rubberband algorithm to compute (note: approximately, defined by the chosen accuracy ε) the ESP.

For the input polyhedron we assume that it is type 2. For example, the Schönhardt polyhedron as shown in Fig. 2 is type 2, but it might be rotated such that the resulting polyhedron is not type 2 anymore.

Algorithm 2 (main algorithm)
Input: *Two points p and q in Π; sets S and V of faces and vertices of Π, respectively.*
Output: *The set of all vertices of an approximate shortest path, starting at p and ending at q, and contained in Π.*

```
1: Compute $V' = \{v : p_z < v_z < q_z \wedge v \in V\}$.
2: Sort $V'$ according to the $z$-coordinate.
3: We obtain $V' = \{v_1, v_2, \ldots, v_{k'}\}$ with $v_{1z} \leq v_{2z} \leq \ldots \leq v_{k'z}$.
4: Partition $V'$ into pairwise disjoint subsets $V_1$, $V_2$, ..., and $V_k$ such that
   $V_i = \{v_{i1}, v_{i2}, \ldots, v_{in_i}\}$, with $v_{ij_y} = v_{ij+1_y}$, for $j = 1, 2, \ldots, n_i - 1$, and
   $v_{i1_y} < v_{i+1 1_y}$, for $i = 1, 2, \ldots, k - 1$. [That is, this step partitions the set $V'$ into
   some subsets such that the points in the same subset have an identical $y$-coordinate.]
5: Let $u_i = v_{i1}$, where $i$ = 1, 2, ..., $k$.
6: Let $V'' = \{u_1, u_2, \ldots, u_k\}$ (then we have that $u_{1z} < u_{2z} < \ldots < u_{kz}$).
7: for each $u_i \in V''$ do
8:     Apply Procedure 1 for computing $V_{u_i}$ (i.e., a sequence of vertices of the critical
       polygon $P_{u_i}$).
9: end for
10: Let $S_{step} = \{P^\bullet_{u_1}, P^\bullet_{u_2}, \ldots, P^\bullet_{u_k}\}$.
11: Let $P = \{p\} \cup V'' \cup \{q\}$.
12: Apply Algorithm 1 on inputs $S_{step}$ and $P$, for computing the shortest path $\rho(p, q)$
    inside of $\Pi$.
13: Convert $\rho(p, q)$ into the standard form of a shortest path by deleting all vertices
    which are not on any edge of $\Pi$ (i.e., delete $p_i$ if $p_i$ is not on an edge of $P_{u_i}$).
```

Theorem 2. *The solution obtained by Algorithm 2 is an approximate global solution to the 3D ESP problem.*

Proof. Let $X = \Pi_{i=1}^{k} P^\bullet_{u_i}$, where $P^\bullet_{u_i}$ is as defined in Algorithm 2. Let Y be the set of all solutions obtained by Algorithm 2. By Lemmas 1 and 2, and Theorem 1, Algorithm 2 defines a continous function, denoted by f, mapping from X to Y depending on the accuracy ε used in Algorithm 1.

If each $P^\bullet_{u_i}$ is degenerated to a single edge, then there exists a unique solution to the ESP problem; see [5, 13, 15]. Now, let $v = (v_1, v_2, \ldots, v_k) \in Y$. Then v_i is located on an edge of polygon P_{u_i}, which is contained in $P^\bullet_{u_i} \setminus P^\circ_{u_i}$ (the frontier of $P^\bullet_{u_i}$), where i = 1, 2, ..., k; or v_i is located in $P^\circ_{u_i}$ (inside of polygon P_{u_i}), and v_{i-1}, v_i and v_{i+1} are colinear. Thus, Y is a finite set.

Now, we prove that Y is a singleton. Otherwise, take $v_0 \in Y$, then we have $f^{-1}(v_0) \subset X$. For each $v \in f^{-1}(v_0)$, as f is a continous function, there exists a sufficiently small open neighborhood (with respect to the usual topology on X) of v, denoted by $N(v,\delta_v)$, such that for each $v' \in N(v,\delta_v)$, $f(v') = v_0$. Thus, $N(v,\delta_v) \subseteq f^{-1}(v_0)$ and $\cup_{v \in f^{-1}(v_0)} N(v,\delta_v) \subseteq f^{-1}(v_0)$.

On the other hand, as $f^{-1}(v_0) = \{v : v \in f^{-1}(v_0)\}$ and $v \in N(v,\delta_v)$, thus we have $f^{-1}(v_0) \subseteq \cup_{v \in f^{-1}(v_0)} N(v,\delta_v)$. Therefore, $f^{-1}(v_0) = \cup_{v \in f^{-1}(v_0)} N(v,\delta_v)$. As $N(v,\delta_v)$ is an open set, $f^{-1}(v_0)$ is an open set as well. Let

$$f^{-1}(v_0) = \cup_{i=1}^{k} S_i$$

where S_i is an open subset of $P^\bullet_{u_i}$, i = 1, 2, ..., k. Recall that $f^{-1}(v_0) \subset X$, so there exists a S_i such that $\emptyset \subset S_i \subset P^\bullet_{u_i}$.

Without loss of generality, suppose that $\emptyset \subset S_1 \subset P^\bullet_{u_1}$. This implies that there exists a point $(x_0, y_0) \in P^\bullet_{u_1}$ such that $\emptyset \subset S_1|_{x_0} \subset P^\bullet_{u_1}|_{x_0}$.[4] Now, $S_1|_{x_0}$ is a nonempty open subset of $P^\bullet_{u_1}|_{x_0}$. $S_1|_{x_0}$ is a union of a countable number of open intervals or half open intervals (Proposition 5.1.4, [14]).

Thus, there exists a point $w_1 \in P^\bullet_{u_1}|_{x_0} \setminus S_1$ such that, for each positive ε_1, there exists a point $w_1' \in N(w_1, \varepsilon_1) \cap S_1$ [again, $N(w_1, \varepsilon_1)$ is an open neighborhood with respect to the usual topology on $P^\bullet_{u_1}$]. Therefore, there exists a point $v_1 \in X \setminus f^{-1}(v_0)$ such that, for each positive ε_1, there exists a point $v_1' \in N(v_1, \varepsilon_1) \cap f^{-1}(v_0)$. This contradicts that f is a continous function on X. Thus, Y is a singleton. □

4 Time Complexity

Lemma 3. *Procedure 1 can be computed in $\mathcal{O}(|V_v||E(S_v)| + |S|)$ time.*

Proof. Step 1 can be computed in $\mathcal{O}(|S|)$ time. Step 4 can be computed in time $\mathcal{O}(|S_v|)$. Step 10 can be computed in $\mathcal{O}(|E(S_v)|)$ time. Step 13 can be computed in time $\mathcal{O}(|E(S_v)|)$. Steps 14 and 17 can be computed in time $\mathcal{O}(|S_v|)$. Steps 15 and 18 only needs $\mathcal{O}(|V_v|)$ time. It follows that Steps 5–21 can be computed in time $\mathcal{O}(|E(S_v)|)$ because of $|V_v| \leq |S_v| \leq |E(S_v)|$.

Step 22 can be computed in time $\mathcal{O}(|S_v|)$. Thus, each iteration (Steps 3–24) can be computed in time $\mathcal{O}(|E(S_v)|)$ because of $|S_v| \leq |E(S_v)|$. Step 26 only needs $\mathcal{O}(|V_v|)$ time.

Altogether, Procedure 1 has a time complexity in $\mathcal{O}(|V_v||E(S_v)| + |S|)$. □

Lemma 4. *Algorithm 1 can be computed in $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V_{v_j}|)$ time.*

Proof. Steps 2–5 can be computed in time $\mathcal{O}(k)$. By Theorem 1, Steps 6–15 can be computed in $\mathcal{O}(|V_{v_j}|)$ time, where V_{v_j} is as in Algorithm 1, for j = 1, 2, ..., k. Thus, each iteration of Algorithm 1 can be computed in $\mathcal{O}(\sum_{j=1}^{k} |V_{v_j}|)$ time. Therefore, Algorithm 1 can be computed in $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V_{v_j}|)$ time. □

Theorem 3. *Algorithm 2 can be computed in*

$$\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V_{u_j}|) + \mathcal{O}(\sum_{j=1}^{k} |V_{u_j}||E(S_{u_j})| + |S| + |V| \log |V|)$$

where the second term is the time for preprocessing.

[4] $S|_{x_0} = \{(x_0, y) : (x, y) \in S \wedge x = x_0\}$

Proof. Step 1 can be computed in $\mathcal{O}(|V|)$ time, and Step 2 in $\mathcal{O}(|V'|log|V'|)$ time. Steps 4 and 5 can be computed in $\mathcal{O}(|V'|)$ time and Step 6 in $\mathcal{O}(|V''|) = \mathcal{O}(k)$ time. By Lemma 3, Steps 7–9 require a computation time in $\mathcal{O}(\sum_{j=1}^{k} |V_{u_j}||E(S_{u_j})| + |S|)$.

Step 10 can be computed in $\mathcal{O}(|V''|) = \mathcal{O}(k)$. By Lemma 4, Step 12 has a time complexity in $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V_{u_j}|)$. Step 13 can be computed in $\mathcal{O}(\sum_{j=1}^{k} |V_{u_j}|)$ time. Therefore, Algorithm 2 can be computed in time $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V_{u_j}|)$ if excluding to count the preprocessing time $\mathcal{O}(\sum_{j=1}^{k} |V_{u_j}||E(S_{u_j})| + |S| + |V| \log |V|)$. □

Corollary 1. *Algorithm 2 can be computed in*

$$\kappa(\varepsilon) \cdot \mathcal{O}(M|V|) + \mathcal{O}(M|E| + |S| + |V| \log |V|)$$

where the second term is the time for preprocessing, and $M = \max\{|V_{u_j}| : j = 1, 2, \ldots, k\}$.

5 Examples: Three NP-complete or NP-hard Problems

We apply Algorithms 1 and 2 for the approximate solution of hard problems, characterized below as being NP-complete or NP-hard. Let p, $q \in \Pi$ such that $p_z < q_z$. Let $V_{pq} = \{v : p_z < v_z < q_z \wedge v \in V\}$, where V is the set of all vertices of Π. For doing so, we are allowing for input polyhedra different from the bounded type-2 polyhedra so far, but only input ployhedra which allow to use those algorithms without any further modification.

We consider unbounded polyheda (which also satisfy the type-2 constraint), and, thus, generalized critical polygons.

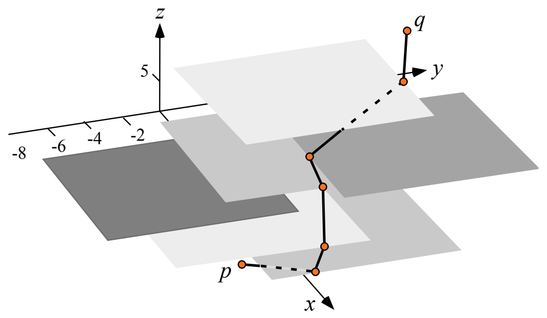

Fig. 3. A path from p to q which does not intersect any of the shown rectangles at an inner point.

Example 1. Let Π be a simply connected polyhedron such that each critical polygon is the complement of an axis-aligned rectangle. Following Section 4, the Euclidean shortest path between p and q inside of Π can be approximately computed in $\kappa(\varepsilon) \cdot \mathcal{O}(|V_{pq}|)$ time. Therefore, the 3D ESP problem can be approximately solved efficiently in such a special case; see Fig. 4 for experiments up to 550 randomly generated rectangles in 3D space. Finding the exact solution is very hard (NP-complete!) because of the following

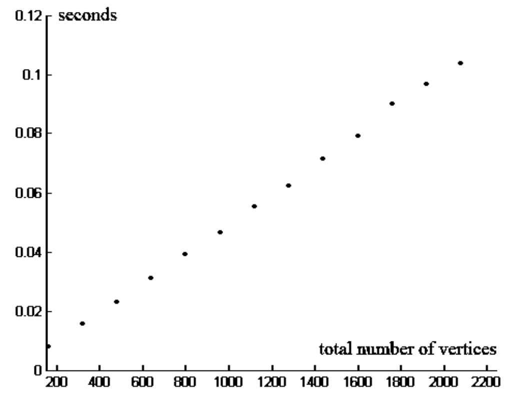

Fig. 4. Calculation of approximate shortest paths for various sets of stacked rectangles: measured time for $\varepsilon = 10^{-10}$ versus total number of vertices (Java, run under Matlab 7.0.4, Pentium 4).

Theorem 4. ([11], Theorem 4*) It is NP-complete to decide wether there exists an obstacle-avoiding path of Euclidean length at most L among a set of stacked axis-aligned rectangles. The problem is (already) NP-complete for the special case that the axis-aligned rectangles are all q-rectangles of types 1 or 3.*

Example 2. Let Π be a simply connected polyhedron such that each critical polygon is the complement of a triangle. Following Section 4, the Euclidean shortest path between p and q inside of Π can be approximately computed in $\kappa(\varepsilon) \cdot \mathcal{O}(|V_{pq}|)$ time. Finding the exact solution is very hard (NP-hard!) because of the following

Theorem 5. ([4]) *It is NP-hard to decide whether there exists an obstacle-avoiding path of Euclidean length at most L among a set of stacked triangles.*

Example 3. Let $\mathcal{S}$ be a stack of k horizontal or vertical strips. The Euclidean shortest path among $\mathcal{S}$ can be approximately computed in $\kappa(\varepsilon) \cdot \mathcal{O}(k)$ time. Finding the exact solution is very hard (NP-complete!) because of the following

Theorem 6. ([11], Theorem 5*) It is NP-complete to decide whether there exists an obstacle-avoiding path of Euclidean length at most L among a finite number of stacked horizontal and vertical strips.*

Example 4. Let $\mathcal{S}$ be a stack of k terrain-like axis-parallel rectangles. The Euclidean shortest path among $\mathcal{S}$ can be approximately computed in $\kappa(\varepsilon) \cdot \mathcal{O}(k)$ time. The best known algorithm for finding the exact solution has a time complexity in $\mathcal{O}(k^4)$ due to

Theorem 7. ([11], Theorem 6*) Let $\mathcal{S}$ be a stack of k terrain-like axis-parallel rectangles. The Euclidean shortest path among $\mathcal{S}$ can be computed in $\mathcal{O}(k^4)$ time.*

6 Conclusions

This paper described an algorithm for solving the 3D ESP problem when the domain Π is a type-2 simply connected polyhedron; the algorithm has a time complexity in $\kappa(\varepsilon) \cdot \mathcal{O}(M|V|) + \mathcal{O}(M|E| + |S| + |V| \log |V|)$ (where $\mathcal{O}(M|E| + |S| + |V| \log |V|)$ is the time for preprocessing). It was also shown that the algorithm approximately solves three NP-complete or NP-hard problems in time $\kappa(\varepsilon) \cdot \mathcal{O}(k)$, where k is the number of layers in the given stack of polygons.

Our algorithm has straightforward applications on ESP problems in 3D imaging (where proposed solutions depend on geodesics), or when 'flying' over a polyhedral terrain. The best result so far for the latter problem was an $\mathcal{O}((n/\varepsilon)(\log n)(\log\log n))$ algorithm for computing a $(2^{(p-1)/p} + \varepsilon)$-approximation to the L_p-shortest path above a polyhedral terrain.

As there does not exist an algorithm for finding exact solutions to the general 3D ESP problem (see Theorem 9, [3]), our method defines a new opportunity to find approximate (and efficient!) solutions to the discussed classical, fundamental, hard and general problems.

References

1. P. K. Agarwal, R. Sharathkumar, and H. Yu. Approximate Euclidean shortest paths amid convex obstacles. In Proc. *Annu. ACM-SIAM Symp. Discrete Algorithms*, 283–292, 2009.
2. L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In Proc. *Annu. ACM Sympos. Theory Comput.*, pages 286–295, 2000.
3. C. Bajaj. The algebraic complexity of shortest paths in polyhedral spaces. In Proc. *Allerton Conf. Commum. Control Comput.*, pages 510–517, 1985.
4. J. Canny and J.H. Reif. New lower bound techniques for robot motion planning problems. In Proc. *IEEE Conf. Foundations Computer Science*, pages 49–60, 1987.
5. J. Choi, J. Sellen, and C.-K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In Proc. *Annu. ACM Sympos. Computational Geometry*, pages 350–359, 1995.
6. K. L. Clarkson. Approximation algorithms for shortest path motion planning. In Proc. *Annu. ACM Sympos. Theory Comput.*, pages 56–65, 1987.
7. S. Har-Peled. Constructing approximate shortest path maps in three dimensions. In Proc. *Annu. ACM Sympos. Comput. Geom.*, pages 125–130, 1998.
8. F. Li and R. Klette. Rubberband algorithms for solving various 2D or 3D shortest path problems. In Proc. *Computing: Theory and Applications*, The Indian Statistical Institute, Kolkata, pages 9 - 18, IEEE, 2007.
9. Y. A. Liu. and S. D. Stoller. Optimizing Ackermann's function by incrementalization. In Proc. *ACM SIGPLAN Sympos. Partial Evaluation Semantics-Based Program Manipulation*, pages 85–91, 2003.
10. J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, editors). pages 633–701, Elsevier, 2000.
11. J. S. B. Mitchell and M. Sharir. New results on shortest paths in three dimensions. In Proc. *Annu. ACM Sympos. Comput. Geom.*, pages 124–133, 2004.
12. C. H. Papadimitriou. An algorithm for shortest path motion in three dimensions. *Inform. Process. Lett.*, **20**:259–263, 1985.
13. M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, **15**:193–215, 1986.
14. B. G. Wachsmuth. Interactive real analysis. See `http://web01.shu.edu/projects/reals/topo/index.html` (last visit: May, 2009).
15. C.-K. Yap. Towards exact geometric computation. *Computational Geometry: Theory Applications.*, **7**:3–23, 1997.
16. Y. Wang, B. S. Peterson, and L. H. Staib. 3D brain surface matching based on geodesics and local geometry. *Computer Vision Image Understanding*, **89**:252–271, 2003.
17. H. Z. Zadeh. Flying over a polyhedral terrain. *Inform. Process. Lett.*, **105**:103–107, 2008.