

# Expansion-Based Algorithms for Finding Single Pair Shortest Path on Surface

Ke Deng and Xiaofang Zhou

School of Information Technology and Electrical Engineering  
University of Queensland, Brisbane, QLD 4072 Australia  
<dengke, zxf@itee.uq.edu.au>

**Abstract.** Finding single pair shortest paths on surface is a fundamental problem in various domains, like Geographic Information Systems (GIS) 3D applications, robotic path planning system, and surface nearest neighbor query in spatial database, etc. Currently, to solve the problem, existing algorithms must traverse the entire polyhedral surface. With the rapid advance in areas like Global Positioning System (GPS), Computer Aided Design (CAD) systems and laser range scanner, surface models are becoming more and more complex. It is not uncommon that a surface model contains millions of polygons. The single pair shortest path problem is getting harder and harder to solve. Based on the observation that the single pair shortest path is in the locality, we investigate in this paper efficient methods by excluding part of the surface model without considering them in the search process. Three novel expansion-based algorithms are proposed, namely, Naïve algorithm, Rectangle-based Algorithm and Ellipse-based Algorithm. Each algorithm uses a two-step approach to find the shortest path. (1) compute an initial local path. (2) use the value of this initial path to select a search region, in which the global shortest path exists. The search process terminates once the global optimum criteria are satisfied. By reducing the searching region, the performance is improved dramatically in most cases.

## 1 Introduction

The algorithms for finding shortest path on polyhedral surface are the fundamental in a large variety of applications and research areas. In robotic systems, finding an optimal collision-free path for robots in a given space [8] is known as path planning. With the path planning technology, the virtual camera in virtual endoscopy can be guided to pass through an anatomical object for comprehensive disease diagnosis [5]. In 3D GIS systems, many applications require reporting shortest path between two entities over terrain as a basic function. In bioinformatics, by searching and comparing the shortest paths between objects, tasks like gene recognition and protein structure analysis could be conducted. In addition, another emerging interesting research area is the surface nearest neighbor query in spatial database, which is essential in applications such as emergency response, wildlife behavior monitoring and battlefield surveillance.

Many algorithms have been proposed to find the shortest path on surfaces. These algorithms usually work on polyhedral surfaces which are represented as meshes consisting of triangular faces. Meshes are usually generated by sampling and triangulating uniformly distributed points on the surface of the target object. For example, in GIS, the terrain surface is normally represented as the triangular network for visualization or other purpose. Such network can be regular or irregular. In principle, existing shortest path algorithms try to solve three major problems: all pairs, single source and single pair. While the all pairs problem is to find the shortest paths between any pair of vertices of the triangulated mesh, the single source problem is to find the shortest paths from the fixed source point to any other vertices. The single pair problem is to find the shortest path from the fixed source point to some specified destination point. In this paper, we will address the single pair problem on the mesh surface. Efficient algorithm for single pair problem has special interest in applications like surface KNN and path planning.

Among those algorithms, some of them [7–9] first pre-process the polyhedral surface into subdivisions, whereby the exact shortest path from the fixed source point to a given query point can be reported quickly. Chen & Han proposed an algorithm [2, 3], which is one of the best and the only feasible algorithm today. Without any pre-processing, Chen & Han algorithm can find the exact shortest path in time complexity  $O(n^2)$ . To search the global optimal shortest path, all these algorithms need to traverse the entire polyhedral surface. On the other hand, advances in areas such as GPS, CAD systems and laser range scanner are producing growing larger polyhedral surface datasets. A surface containing millions of polygons are not uncommon. Typically, processing the geometric data is very computational expensive and memory hungry. This makes the single pair problem extremely costly and less practical even with high profile computer system. It is urgent to develop approaches to alleviate the situation.

Some works have been done. Takashi et. al. [4] proposed an algorithm for the approximate shortest path with selective refinement technology on polyhedral surface. Dinesh et. al. [6] developed an algorithm based on multiresolution technology. Both of them improved the performance of single-pair problem notably. However, their algorithms still need to traverse the entire surface. The initiative behind these methods is to reduce the computational complexity by reducing the problem complexity. Based on the similar concept, we develop some new algorithms.

We observed that the local shortest path connecting two points inside a certain region is also global optimal. For example, the shortest path between Brisbane and Sydney cannot be the one that pass through Melbourne. Motivated by this observation, our proposed algorithms use a two-step approach to find the shortest path. First, we need to find an initial local path. Second, we use the value of this initial path to select a search region, in which the global shortest path exists. In this paper, three algorithms: Naïve algorithm, Rectangle Border algorithm (RB) and Ellipse Border algorithm (EB) are developed. To our knowledge, no such algorithms have been proposed before. The Naïve

algorithm is conceptually simple and works as the benchmark to evaluate the other two algorithms. In RB algorithm, we offer two alternatives terms of the way of testing global optimum criteria and extension methods, namely Rectangle Border Direct Algorithm (RBD) and Rectangle Border Indirect Algorithm (RBID). In EB algorithm, heuristics is used to estimate the initial length with the consideration of surface roughness.

Comparing with others, our algorithms have four advantages. First, they do not require traversing the entire polyhedral surface. By narrowing down the search region to a selected area, the processing time can be significantly reduced, hence improve the performance of existing algorithms. Second, it is easy to graft our technique on top of an existing algorithms. They can work with existing shortest path algorithms, such as Chen & Han algorithm and Takashi et. al. approximate algorithm. Third, our algorithms are simple and easy to implement. Last, the idea used in our algorithm is novel. Two steps in our proposed algorithms are independent each other. Several different methods are presented for each step in this paper. This provides a great flexibility to select an optimum combination for the best performance in some context.

The rest of the paper is organized as follows. In Sect. 2, a brief overview of related works is presented. Section 3 describes our three proposed algorithms in detail. Performance study is discussed in Sect. 4. Finally, conclusions are given in Sect. 5.

## 2 Related Work

Shortest path problems have been studied for the last two decades. For a given polyhedral surface and a point, the surface can be pre-processed to produce a data structure. The shortest path from the specified point to any query point can be reported with the aid of this structure. [7, 8] Mitchell et. al. [7] improved this wavefront propagation method and named it as “continuous Dijkstra”. In “continuous Dijkstra”, a “signal” is propagated from source to the rest of the surface. Each time a point records the shortest distance from source when it receives this “signal” and propagate it further. The pre-processing operation can be done in time  $O(n^2 \log n)$ , then the shortest path is reported in time  $O(k + \log n)$ , where  $k$  is the number of faces crossed by the path and  $n$  is the total number of vertices. Later on, Chen & Han [2, 3] proposed an different algorithm other than “continuous Dijkstra” technique. With this algorithm, the shortest path between two vertices can be reported in time  $O(n^2)$  without pre-processing the surface into subdivision. A sequence tree structure is built during the process of unfolding all the faces of the polyhedral surface. Each node of the tree represents a set of shortest path which all have the same edge sequence and angularly contiguous at the source. Chen & Han avoid the exponential trap by keeping the “one single one split” property throughout tree structure construction. Chen & Han algorithm is currently one of the fastest and only feasible method to search the exact shortest path on polyhedral surface. Both of them could deal with the non-convex polyhedral surface. Clearly, the surface partitioning facilitates the

quick report of the shortest path. But it does not favor all cases. For example, in the battle surveillance system and wildlife behavior monitoring system, only few objects are our real interests. And the objects including source itself are always in moving situation. In such circumstance, the costly surface subdivision for moving source point will not be an attractive choice.

With the attempt to balance the cost in time and approximation accuracy, various methods are developed to find approximate shortest path [10–13, 4]. Takashi et. al. [4] proposed an algorithm for single-pair problem on a polyhedral surface, which mainly used Dijkstra’s algorithm and is based on selective refinement of the discrete graph of the polyhedron. In their method, Dijkstra’s algorithm is iteratively used to narrow down the region in which the shortest path can exist. This approximate algorithm can calculate shortest path within 0.4% accuracy to roughly 100 – 1000 times faster comparing with Chen & Han algorithm in experiment. Away from the approximate algorithms, Dinesh et. al. [6] developed an approach to solving the single pair problem for the robotic system motion planning. Their approach computes a multiresolution representation of the terrain using wavelets, and hierarchically plans the path through sections, which are well approximated on coarser levels and relatively smooth. These two approaches described above concentrate on the single pair shortest path problem and they try to narrow down the search area and thus simplify the problem.

However, all these algorithms mentioned in this section need to traverse the entire surface to find the single pair shortest path. Moreover, the approximate method cannot report exact shortest path. Dinesh’s algorithm does not offer a comprehensive experiment result about their algorithms performance. In contrast, one contribution of our algorithms in this paper is that our approach does not require navigating the entire polyhedral surface to solve single pair shortest path problem. The essential method we used is to select the search region in a carefully designed range and test the local shortest path using the global optimal criteria. The shortest path search terminates once the global optimal criteria are satisfied.

### 3 Algorithms

In this section, three approaches to obtain the shortest path on the polyhedral terrain surface are detailed, where surface is represented as meshes consisting of triangular planar. Let  $V = \{v_i | i = 1 \dots n\}$  be a set of vertices on the polyhedral terrain surface  $G^1$ , where  $v_i$  denotes each from  $s$  to  $d$  is represented by  $p_{sd}$ . The  $xy$ -plane in three dimensional coordinate system is defined as  $Plane_{xy}$ . In this paper, because we work on the polyhedral terrain surface, only  $xy$ -plane will be discussed. We also denote the straight line between points  $s$  and point  $d$  as  $sd$  and the orthogonal projections of  $s$  and  $d$  in  $xy$ -plane are  $s'$  and  $d'$ , there is

---

<sup>1</sup> A polyhedral terrain is a polyhedral surface that every vertical line intersects it in at most one point. The polyhedral terrain surface may be convex or non-convex, with or without boundary.

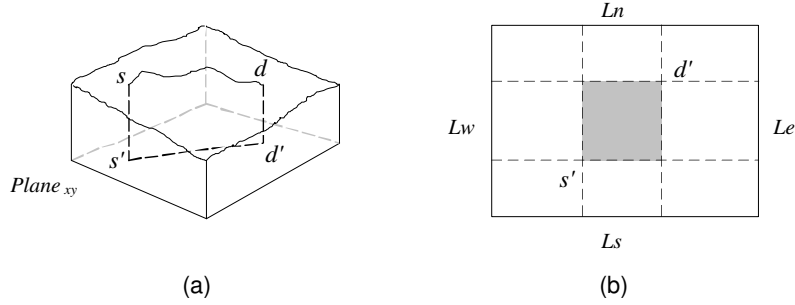
following relation  $s'd' = sd \times \cos\theta$ . Here,  $\theta$  is the angle between  $sd$  and  $s'd'$ . In geometry, Euclidean distance corresponding to the straight line between two points is the shortest, that is, any path connecting two points on the surface are always longer than or equal to the line segment of their projection on coordinate plane. This concept is applied in the rest of the paper.

To compute the shortest path more efficiently, we apply geometric rules on polyhedral terrain surface to identify the maximum possible region, in which the global shortest path exists, and can only be exist within this region. Our algorithms show that any path going beyond the border of the selected region will not be the shortest one.

We develop three algorithms in this paper: Naïve algorithm, RB and EB to find the global optimal shortest path. Each algorithm contains two steps: (1) compute the initial path connecting  $s$  and  $d$ . This step should commit two objectives, computing at low cost and returning optimum initial path. However, optimum initial path is achieved at the expense of computing cost, and vice versa. we design the first step to obtain the initial path at low cost in Naïve and RB algorithm. Whereas in EB algorithm, we design this step by considering the trade-off between those two objectives. (2) select the region that encloses the global shortest path. We offer four methods to select the region as will be shown in the following subsections. Technically, the range of the selected region is determined by the length of the initial path. Once we select the search region, we use Chen & Han algorithm to search the shortest path. Currently their algorithm is the only simple and feasible method of computing the shortest path, yet other algorithms could also be applied. When the shortest path is found in the selected region, the global optimum criteria must be tested to certify its optimum in global. Global optimum criteria are a set of rules which is closely related with the methods in each algorithm. If the global optimum criteria are satisfied, the current shortest path is also global optimum. Otherwise, the shortest path will be search again in the larger area until the criteria are satisfied.

### 3.1 Naïve Algorithm

**Step one** In this algorithm, The method used to search the initial path is very fast. The idea is that any path connecting  $s$  and  $d$  on the surface can be the initial path. For the sake of simplicity, we choose one path whose projection in  $xy$ -plane is the line segment  $s'd'$ , see Fig. 1 (a). All the triangles that the initial path  $p_{sd}$  passes through form a triangle sequence. By computing every component of the initial path in the triangle sequence, the length of the initial path is the sum of these segments. This step is done in time  $O(n)$ , where  $n$  is the number of triangles in the minimum bounding rectangle (MBR). In Fig. 1 (b), the gray rectangle is the MBR in  $xy$ -plane where  $s'$  and  $d'$  are its vertices. Clearly, the initial path  $P_{sd}$  calculated by this way is not local optimum, because it can not satisfy the requirement of the shortest path. Generally, the shortest path should be built on local optimum that the path enter and leave the edge in the same angle, and it is a straight line after unfolding each triangle planar.



**Fig. 1.** Naïve algorithm (a) initial path estimation (b) Selected region with new borders

**Step two** Once we calculate the initial path, we can select the search region by extending from the MBR used for initial path. In each of the four directions, a new border  $L_w$ ,  $L_e$ ,  $L_n$  and  $L_s$  should be identified, as being depicted in Fig. 1 (b). The sum of the vertical distance from point  $s'$  to line  $L_w$  and the vertical distance from point  $d'$  to line  $L_w$  should be equal to the length of the initial path, so do other three directions. In this case, any path connecting  $s'd'$  in  $xy$ -plane is longer than the initial path if it goes beyond the new selected region bounded by solid lines. In other words, any path connecting  $s$  and  $d$  must be inside the new selected region. passing through any point whose projection is outside the new selected border, the path is longer than the initial path.

This algorithm uses an easy way to search the initial path. The advantage is its low cost. However, because the initial path is in a pre-determined direction along the surface, it may be happen to pass the roughest terrain region. The length of the initial path may turn out to be severely skewed. In turn, the long initial path will lead to an unnecessary large search region in step 2. In some cases, the cost saved in initial step will be consumed by the shortest path search computation in step 2.

### 3.2 Rectangle Border Algorithm

We offer two alternatives under this algorithm in terms of the way of testing global optimum criteria and extension methods, namely RBD and RBID. Both of them using the same approach in step one but differ in step two.

**Step one** Due to the inherent weakness of the initial path searching method in Naïve algorithm, the search region in step two may be unnecessarily large. By finding the local optimum initial path, this problem can be avoided. We draw a rectangle region around points  $s'$  and  $d'$ . This rectangle is a bit wider than the MBR in Naïve algorithm. The surface bounded by this rectangle are used to search the local shortest path connecting  $s$  and  $d$ . Any point on the bounded surface has its projection within the rectangle in  $xy$ -plane. If we use Chen &

Han's algorithm as the basic shortest path search algorithm, the local shortest path is found in time complexity  $O(n^2)$ , where  $n$  is the number of triangles in the search region. Even though the cost in this step is much higher than  $O(n)$  in Naïve algorithm, this method has its own advantages: (1) the output initial path is local optimal that could optimize the search region in step two. (2) the current local shortest path may also be the global optimum, yet we need to apply the global optimal criteria for this.

## Step two

*Rectangle Border Indirect Algorithm:* The local optimum initial path found in step one may or may not be the global optimum, therefore we first apply global optimal criteria test against the output. In this method, the global optimum criteria make use of the rectangle border as a special component of the surface. During the searching process, the path can traverse along the surface, or if the path reaches the border, it can take the shortcut along the border.

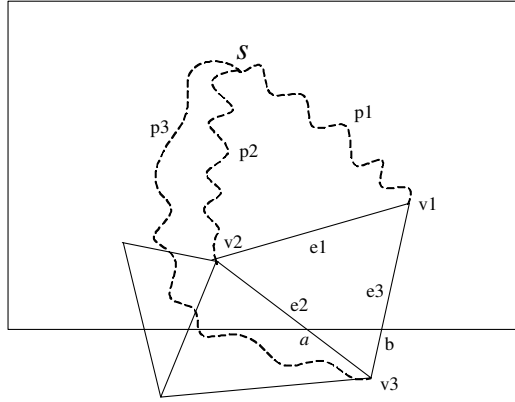
Chen & Han's algorithm can directly report the shortest path between vertices. However, it can not easily find shortest path for any point inside the triangle on the polyhedral surface. If the path from the source point to any point inside the triangle is required, this point needs to be triangulated as a vertex. To simplify the problem, the distance to any point inside a triangle can be defined in a certain range. This is the Lemma 1.

**Lemma 1.** *Let  $e_1$ ,  $e_2$  and  $e_3$  be the edges of a triangle on the surface, and  $p_1$ ,  $p_2$  and  $p_3$  be the shortest paths from the source point  $s$  to the three vertices  $v_1$ ,  $v_2$ ,  $v_3$ . If  $e_1 > e_2 > e_3$  and  $p_1 > p_2 > p_3$ , the shortest path  $p_r$  from  $s$  to any point  $r$  inside the triangle will be  $p_1 + e_1 \geq p_r \geq p_3 - e_1$*

*Proof.* If any point inside this triangle does not follow above relations, its shortest path  $p_r$  is less than  $p_3 - e_1$ , then we have the relation  $p_r + e_1 < p_3$ . Because point  $r$  and three vertices are on the same planar, and  $e_1$  is the longest edge, the distance  $dist_r$  from point  $r$  to the vertices must be shorter than  $e_1$ . That is  $dist_r < e_1$ . So  $p_r + dist_r < p_r + e_1 < p_3$ . This indicates that there is a path  $p_r + dist_r$ , which is less than the known shortest path  $p_3$ . This contradicts the fact that  $p_3$  is the shortest path. Therefore,  $p_1 + e_1 > p_r$  can be proved.  $\square$

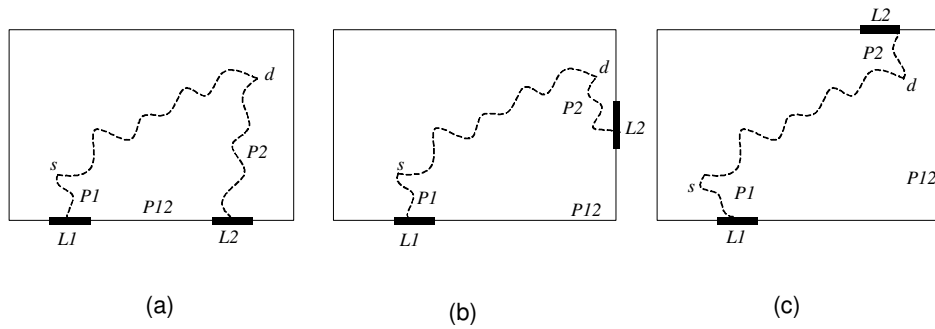
The rectangle border is cut into line segments by the triangles whose projections in  $xy$ -plane intersect the border. Then, the border can be treated as line segments instead of points. For example, in Fig. 2,  $ab$  is one of the line segments. If all border-touched path is longer than or equal to the local shortest path, the local shortest path must be the global optimum. In this case, global optimum criteria are satisfied.

**Theorem 1.** *Given a triangulated surface  $G$ , the local shortest path connecting points  $s$  and  $d$  inside a rectangle region is represented as  $p_{sd}$ . Let  $p_1$  be the shortest path from  $s$  to line segment  $L_1$  and  $p_2$  be the shortest path from  $s$  to line segment  $L_2$ . See Fig. 3. The path along the border connecting the two line*



**Fig. 2.** RBID shortest path range for non-vertex points

segments is  $p_{12}$ , which value is the shortest in space. If for all pairs of border segments:  $p_{sd} \leq p_{12} + p_1 + p_2$ , the global optimum criteria are satisfied, that is,  $p_{sd}$  is the global shortest path.  $\square$

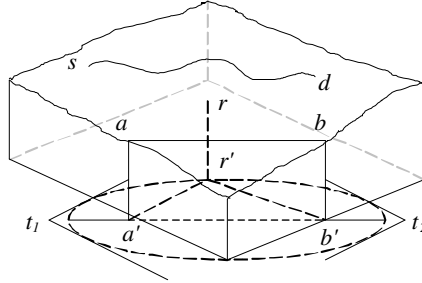


**Fig. 3.** Three situations of global optimum criteria

Because all the border line segments must satisfied the  $p_{sd} \leq p_{12} + p_1 + p_2$ , there are three situations need to be considered, see Fig. 3. According to Lemma 1, the range of  $p_1$  and  $p_2$  can be estimated with the length of the local optimum shortest paths to the three vertices. The path connecting  $L_1$  and  $L_2$  along the border is  $p_{12}$  and it must be the shortest in space. In Fig. 3 (a),  $p_{12}$  is a straight line segment on the border and it is the shortest. In Fig. 3 (b) and (c), the border is not a straight line. If  $p_{12}$  takes the value of the border length, it

cannot guarantee that  $p_{12}$  is the shortest in space. In such situation, we use the Euclidean distance to replace  $p_{12}$ . If  $p_{sd} > p_{12} + p_1 + p_2$  for some line segments, the global optimum criteria are not satisfied. In this situation, the search region must be extended to obtain more surface information, then search the shortest path in this extended region again. One point to note that, because  $p_1$  and  $p_2$  have difference source  $s$  and  $d$ , we need to search the region twice to find the shortest path.

In Naïve algorithm, the method used to extend the search region gives a too wide extension range. We offer a better extension strategy in this RBID algorithm. When testing the border condition for the global optimum criteria, we have known which pair of border line segments do not satisfy the criteria and to what extent the extension should be. The difference between  $p_{sd}$  and  $p_{12} + p_1 + p_2$  (see Fig. 3) sets the upper bound for range of extension. So, ellipse can be used. In geometry, an ellipse is a set of points in a plane the sum of whose distances from two foci is a constant (we call this constant as ellipse constant in the rest of the paper). For points outside the ellipse, the sum of distance to both foci is greater than the ellipse constant. This feature can be employed to identify better extension range. In Fig. 4, points  $a'$  and  $b'$  are on the border of the initial rectangle which are the orthogonal projection of points  $a$  and  $b$ . In  $xy$ -plane, an ellipse is drawn with  $a'$  and  $b'$  as the foci and the ellipse constant is equal to the difference between  $p_{sd}$  and  $p_{12} + p_1 + p_2$ .  $r$  is a point on the surface and  $r'$  is the projection of  $r$ . According to the nature of ellipse,  $a'r' + b'r'$  is the ellipse constant if  $r'$  happens to be on the ellipse. Obviously, at the same time, the sum of the shortest path  $p_{ar}$  and  $p_{br}$  on the surface is longer than or equal to the ellipse constant. On the other side, if  $r'$  is outside the ellipse,  $p_{ar} + p_{br}$  must be greater than the constant. That is, if  $r'$  is outside the ellipse,  $p_{sd} < p_{sa} + p_{ar} + p_{rb} + p_{bd}$ . So, the ellipse-bounded region is the region that may contain shorter path than initial path. For simplicity, the tangent line of the ellipse is used as the new border for this border line segment pair, as depicted in Fig. 4.



**Fig. 4.** RBID extension strategy

Note that the extension approach above is only for border line segments containing  $a'$  and  $b'$ . To decide the extension range, all the border line segment pairs with  $p_{sd} > p_{12} + p_1 + p_2$  should be considered. Finally the biggest extension value will be chosen as the extension range.

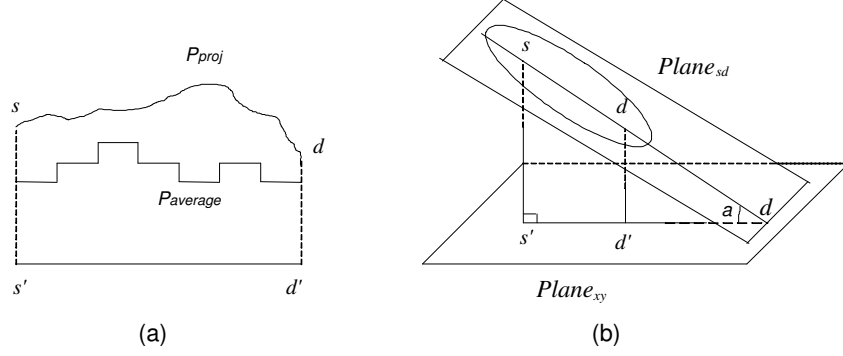
*Rectangle Border Direct Algorithm* : Besides the global optimum criteria testing and extension method discussed above, an alternative approach can be used. After the local shortest path is found, an ellipse is drawn with  $s'$  and  $d'$  as foci and ellipse constant is equal to the length of initial local shortest path. Using this method, we avoid checking all border line segment pairs. This is a straightforward method to test the global optimum criteria. If the range of ellipse is inside the initial rectangle, the global optimum criteria are satisfied. If ellipse is beyond the initial rectangle border, we need to search the shortest path again with the new region defined by the ellipse. Obviously, this method is easy to implement and get even better performance. because it does not require searching the selected region twice.

### 3.3 Ellipse Border Algorithm

**Step one** The approach used in step one of Naïve algorithm can quickly find an initial path, but it is not local optimum. On the contrary, RB algorithm can obtain the local optimum initial path, however with high resource cost. Both of them have advantages and disadvantages. In this section, a more efficient approach is developed to search the initial path by taking into account of surface roughness.

Our approach estimates the initial path  $p_{sd}$  using the following heuristics, see Fig. 5 (a). First, the initial path  $p_{proj}$  is computed using the same method as in step one of Naïve algorithm. The orthogonal projection of  $p_{proj}$  is the line segment  $s'd'$ , which is the projections of  $s$  and  $d$  in  $xy$ -plane. Second, an ellipse with  $s'$  and  $d'$  as its foci is drawn and the ellipse constant equals to  $p_{proj}$ . The surface inside the ellipse is equally partitioned along the direction from  $s$  to  $d$ . The average height  $h_{average}$  of each part is computed. Then, the path  $p_{average}$  along the surface is computed. Next, we compare  $p_{proj}$  and  $p_{average}$ . If  $p_{proj}$  is close to  $p_{average}$ , the estimate value of initial path will be the length of  $p_{proj}$ . The closeness is indicated by ratio  $\delta = \frac{p_{average}}{p_{proj}}$ . If  $\delta > 1.2$ , the estimate value of initial path will be the length of  $p_{average}$ . This implies that  $p_{proj}$  passes through a very rough area.

**Step two** After the initial path is found, the search area needs to be selected. We draw an ellipse in  $sd$ -plane with the initial path as the ellipse constant, and let  $s, d$  as ellipse's foci rather than their projections  $s'$  and  $d'$ , see Fig. 5 (b). The  $sd$ -plane is the plane that contains  $s$  and  $d$ , and intersects  $xy$ -plane at angle  $\omega$ . This choice can further decrease the search region. We let  $a$  equal to the value of the initial path,  $c$  equal to half of the Euclidean distance between  $s$  and  $d$ ,  $b$



**Fig. 5.** Ellipse algorithm (a) Initial path estimation (b) Ellipse search region selection

as the width of the ellipse. There is following relation:

$$b = \sqrt{a^2 - c^2} . \quad (1)$$

Clearly,  $b$  value declines as  $c$  value increases. Because the ellipse is drawn in  $sd$ -plane,  $c$  has greater value than that of in  $xy$ -plane, therefore, search region selected in  $sd$ -plane is smaller than that of in  $xy$ -plane. After the search region is identified, we can compute the shortest path in this area and this shortest path is the global optimum. Following discussion provides the proof.

Given a local shortest path  $p_{short}$  connecting points  $s$  and  $d$  inside an ellipse area on the surface, if and only if the length of the local shortest path  $p_{short}$  is shorter or equal to the ellipse constant, the global optimum criteria are satisfied. That means  $p_{short}$  is the global shortest path. This can be easily proved. Any path  $p_{sd}$  connecting source point  $s$  and destination point  $d$  traverses beyond the ellipse border is longer than the ellipse constant  $Q$ ,  $p_{sd} > Q$ . If the local shortest path  $p_{short} \leq Q$ , it is obvious that  $p_{short} < Q < p_{sd}$ . The global optimum criteria is satisfied and  $p_{short}$  is the global shortest path.

In our case, the ellipse is drawn with length of initial path as the ellipse constant  $Q$ . Because the initial path must be longer than or equal to the local shortest path, thus  $p_{short} \leq initial\ path$ . As  $initial\ path = Q$ , therefore,  $p_{short} \leq Q$ . We can conclude the output local shortest path must also be the global optimum.

## 4 Performance Study

In this section we compare the three algorithms discussed in this paper. The primary performance index used in this section is the processing time, which is measured as the elapsed time from the point of parameters (start and destination points) are set to the global shortest path is reported. As we employed a two-step approach in the proposed algorithms, we test not only the whole processing

time, but also the time spend in each step. According to Kaneva and O'Rourke's experimental report [1], the I/O cost-based measure such as the number of disk pages accessed is not an significant performance indicator because Chen & Han algorithm is typically CPU intensive, thus a small amount of difference in the input data leads the computing time to a superlinear growth. Therefore I/O cost is not measured in our analysis.

#### 4.1 Cost Analysis

The time used to search the global shortest path,  $T_{total}$ , is the sum of time used in step one  $T_{s1}$  and step two  $T_{s2}$ . That is:

$$T_{total} = T_{s1} + T_{s2} . \quad (2)$$

Different methods used in step one of each algorithm are in different time complexity that determine  $T_{s1}$ . Therefore, in Naïve and EB algorithms, step one has time complexity  $O(n)$ , whereas in RB algorithm, step one has time complexity  $O(n^2)$ . In step two, the time  $T_{s2}$  is decided by the time complexity of which the algorithm is chosen. As we choose Chen & Han algorithm as the basic shortest path algorithm in the analysis and experiment,  $T_{s2}$  is in time complexity  $O(n^2)$ . It is clear that our algorithms do not change time complexity of Chen & Han algorithm. Thus the time complexity for all of these three algorithms should be  $O(n^2)$ .

We use the projection of the selected region in  $xy$ -plane,  $Area_{s1}$  and  $Area_{s2}$  to represent the search region in two steps respectively. In the initial step, the number of triangles in  $Area_{s1}$  is represented as  $n_1$ . Similarly, the number of triangles in  $Area_{s2}$  is represented as  $n_2$  for step two. Because  $Area_{s2}$  is extended from  $Area_{s1}^2$ ,  $n_2$  is normally larger than or equals to  $n_1$ , therefore, the processing time is mainly decided by  $n_2$ . Given source point  $s$  and destination point  $d$ ,  $n_2$  is determined by  $Area_{s2}$ .

Let  $a$  be the half length of the initial path,  $c$  be the half of the distance of  $sd$  and  $c'$  be the distance of  $s'd'$ . The area for selected search region in step two of Naïve algorithm can be estimated by:

$$Area_{s2} = 4a^2 . \quad (3)$$

For RB1 algorithm, the area of the selected search region is:

$$Area_{s2} = 4a\sqrt{a^2 - c^2} . \quad (4)$$

For EB algorithm, the area of the selected search region is:

$$Area_{s2} = 4a\sqrt{a^2 - c^2} . \quad (5)$$

$c$  is the projection of  $c'$  in  $xy$ -plane,  $c = c' \times \cos\beta$ .  $\beta$  is the angle between line  $sd$  and  $xy$ -plane. Theoretically, for a given length of initial path, EB algorithm will get the optimum selected search region, followed by RB algorithm and the

Naïve algorithm. However, if the length of initial paths are not the same, the total performance for each algorithm will differ.

Three factors influence the number of triangles in a given region, and consequently affect the processing time. The first factor is the resolution of the triangulated surface. Surface with higher resolution contains more triangles, thus  $n$  is larger for a given area. It has different impacts on  $T_{s1}$  of three algorithms but same for the second step.  $T_{s1}$  for  $O(n^2)$  RB algorithm increases in a quadratic manner, while for  $O(n)$  EB and Naïve algorithm  $T_{s1}$  increases in a linear manner. The second factor is the roughness of the surface which affects the number of triangles as well. The influence of the roughness on processing time follows the same pattern of the first factor. Third factor is the angle  $\varphi$  between the line crossing the  $s'$  and  $d'$  in  $xy$ -plane and the  $x$  axis. Because the projection of the surface region used in initial step is a rectangle which diagonal vertices are  $s'$  and  $d'$ . And the sides of rectangle is parallel with the  $x, y$  axis respectively. The relation between  $\varphi$  and  $Area_{s1}$  is:

$$Area_{s1} = \frac{1}{2} \sin(2\varphi) dist_{s'd'} . \quad (6)$$

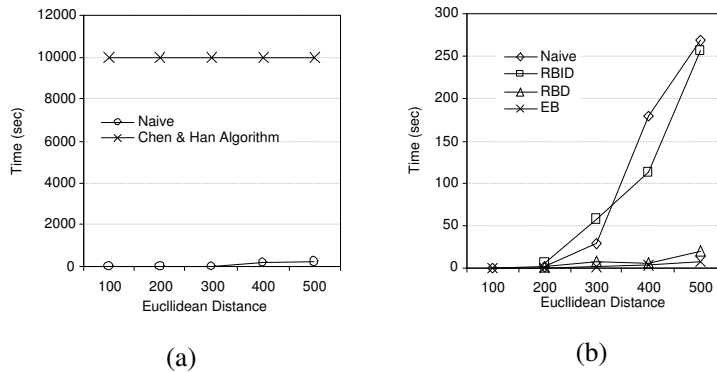
where  $dist_{s'd'}$  is the Euclidean distance between  $s'$  and  $d'$ . The formula shows  $Area_{s1}$  is the maximum when  $\varphi$  is  $\frac{\pi}{4}$ . Angle  $\varphi$  affects step one of all algorithms, yet step two of EB, RBD and Naïve algorithm is independent of this factor. Because of the obvious effect of first two factors on performance, we conduct the experiment concentrating on the last factor.

## 4.2 Experimental Results

In this section, we evaluate the performance of three proposed algorithms for finding the shortest path, namely Naïve, RB and ER algorithm. Two extension methods are tested separately for RB algorithm: RBD and RBID. Both development and testing are done using PC with Pentium IV 2.8G CPU and 512M memory. The operating system is Windows XP Professional. Three algorithms are implemented with C++ compiled with Borland C++ 5.5 compiler. The Microsoft ODBC library is used in the C++ program in order to access the data set. We use a real polyhedral terrain surface data sets in the test, provided by MinCom Pty. Ltd . This data set consists of 10806 triangles and is stored in Oracle Enterprise Edition 9.2. Indexes are created wherever necessary for all the tables. In this experiment, we use the Chen & Han algorithm implemented by Kaneva and O'Rourke [1] to search the shortest path. The performance of proposed algorithms is tested in terms of the processing time.

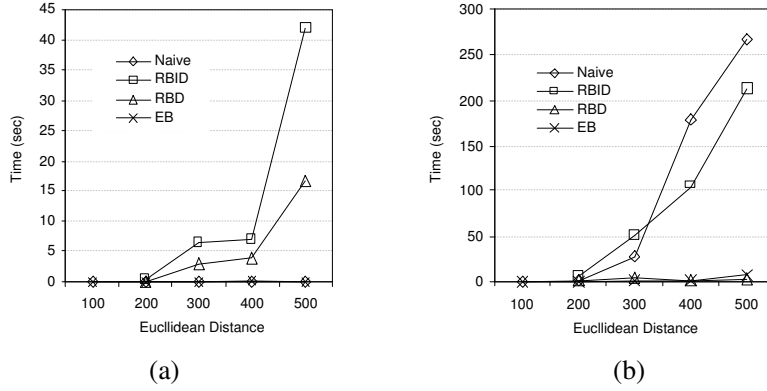
We first test the effectiveness of proposed Naïve, RBD, RBID and EB algorithms in reducing the time cost. The test is conducted by varying  $dist_{s'd'}$  (ranging from 100 to 500, the unit is the same as the coordinate value of the triangles) which is the Euclidean distance between  $s'$  and  $d'$ . At each  $dist_{s'd'}$ , we randomly select ten pairs of source and destination points on the surface. The time usage of the sample data presented is the mean value. Figure 6 (a) shows the time comparison between Naïve algorithm and Chen & Han algorithm. In

order to find the global optimum shortest path, Chen & Han algorithm must compute the entire surface. Instead we select the nonconvex surface with 5175 triangles. Since time spend to report the single pair shortest path by using their algorithm is over 10,000 seconds and is indifferent to the distance changes, we could set this time as a lower time bound for the comparison purpose. Obviously, our proposed algorithm results in a substantial time reduction, i.e. time spend at 500 distance units is less than 500 seconds. Figure 6 (b) illustrates the different time cost among our proposed algorithms with varying  $dist_{s,t}$ . As we can see from the result, the total time used in Naïve and RBID algorithms are very close and increase drastically as the distance becomes larger. Not surprisingly, RBD and EB algorithms outperform the other two. The reason for this difference is explained in Fig. 7.



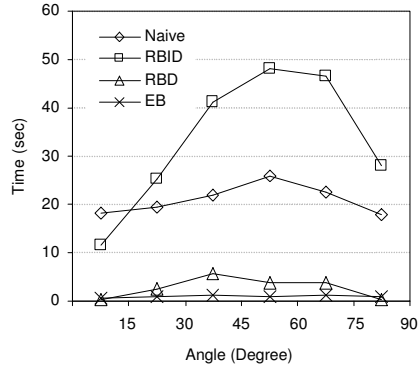
**Fig. 6.** (a)Naive vs. Chen & Han algorithm (b)comparison between proposed algorithms

In Fig. 7, the time cost in each step is drawn separately. Consistent with our cost analysis, where the time complexity in step one for RBID and RBD is  $O(n^2)$ , for EB and Naïve algorithm is  $O(n)$ . Figure 7 (a) shows the time difference among these four algorithms. Both EB and Naïve algorithms have similar time cost as they are base on the same time complexity. One point to note that time spend in RBID is doubled comparing with RBD even though they have the same time complexity. Because in order to test the global optimum criteria, RBID needs to search the selected region twice. Figure 7 (b) reveals an important fact that step two is the dominant factor in the total time cost. The step one has little effect because of its small proportion to the overall time cost. Evidence shows that extension method plays an essential role in the total time cost. Extension methods proposed in EB and RBD are superior to others, therefore, these two algorithms have an outstanding performance. These results strongly support our cost analysis above.



**Fig. 7.** (a) Time cost in step one (b) Time cost in step two

Figure 8 depicts the influence of angle  $\varphi$  on the processing time for all approaches. We randomly select ten pairs of source and destination points for a certain degree, i.e.  $15^\circ, 30^\circ, 45^\circ$  etc. As we cannot obtain the ten pairs of data at the exact same degree, we take the average degree of each group of data. The angle range starts at 0 and ends at  $\frac{\pi}{2}$ . As we discussed in previous section, the angle influences all approaches in step one but rather moderate, and has no effect on Naïve, RBD and EB algorithms in step two except RBID because the methods used to select search region for these three approaches are free of the  $\varphi$  change. It has the maximum impact when  $\varphi = \frac{\pi}{2}$ . The results shown in Fig. 8 support our analysis.



**Fig. 8.** The relation between angle  $\varphi$  and processing time

The above discussions demonstrate that all proposed algorithms can significantly improve the performance of the basic shortest path search algorithms. Among them, EB and RBD approaches are more efficient comparing to RBID the Naïve algorithms in terms of time usage, especially in longer distance.

## 5 Conclusions

With rapid advances in technology, the surface models are becoming more and more complex. The single pair shortest path problem is getting harder and harder to solve. This paper proposed three expansion-based algorithms, namely Naïve algorithm, RB and EB. We also offered two alternatives under RB algorithm in terms of the way of testing global optimum criteria and extension methods: RBD and RBID. In contrast to current algorithms which need to traverse the whole surface to search the global shortest path between a pair of points. This operation is very time-consuming. Motivated by the observation that the single pair shortest path is in the locality, we developed our algorithms based on a two-step strategy: (1) compute an initial local path. (2) use the value of this initial path to select a search region, in which the global shortest path exists. The search process terminates once the global optimum criteria are satisfied. By reducing the searching region, the performance is improved dramatically in most cases. We have shown a comparison between our algorithms and Chen & Han algorithm by using a real data set. Experimental results shows that all of the proposed algorithms can significantly improve the performance of Chen & Han's algorithm, whereas the EB and RBD notably outperform RBID the Naïve algorithm in term of time usage, especially in longer distance. In the future, we plan to integrate the advanced multiresolution technology into our shortest path algorithms.

## Acknowledgments

The work reported in this paper has been supported by an Australian Research Council Discovery Project grant (grant number: DP0345710). We also would like to thank Mincom Pty Ltd for providing access to their terrain data and thank Prof. Hong Shen, Dr. Kai Xu for many helpful discussions.

## References

1. B. Kaneva, J. O'Rourke: An Implementation of Chen & Han's Shortest Paths Algorithm. *In Proc. of the 12th Canadian Conf. on Comput. Geom.*, pages 139-146, 2000.
2. J. Chen, Y. Han. Shortest paths on a polyhedron. In *6th ACM Symposium on Computational Geometry*, Pages 360-369, 1990.
3. J. Chen, Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6:127-144, 1996.

4. T. Kanai, H. Suzuki. Approximate shortest path on polyhedral surface based on selective refinement of the discrete graph and its applications. In *Geometric Modelling and Processing*, pages 241-250, 2000.
5. T. Deschamps, L. D. Cohen. Minimal Paths in 3D Images and Application to Virtual Endoscopy. In *Proc. sixth European Conference on Computer Vision (ECCV'00), Dublin, Ireland, 26th June - 1st July 2000*.
6. D. K. Pai, L. -M. Reissell. Multiresolution Rough Terrain Motion Planning. *IEEE Transactions on Robotics and Automation*, 14 (1): 19-33, February 1998.
7. J. S. B. Mitchell, D. M. Mount, C. H. Papadimitriou. The Discrete Geodesic Problem. *SIAM J. Comput.*, 16: 647-668, 1987.
8. M. Sharr, A. Schorr. On Shortest Paths in Polyhedral Space. *SIAM J. Comput.* 16: 647-668, 1987.
9. C. S. Mata, J. S. B. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. In *Proc. 13th ACM Symp. On Computational Geometry*, pages 265- 273, 1997.
10. M. Lanthier, A. Maheshwari, and J.-R. Sack.. Approximating weighted shortest paths on polyhedral surfaces. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 274-283, 1997.
11. K. R. Varadarajan and P. Agarwal. Approximating shortest paths on an nonconvex polyhedron. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 182-191, Miami Beach, Florida, 20-22 October 1997.
12. S. Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.*, 28:1128-1197, 1999.
13. P. K. Agarwal, S. Har-Peled, M. Sharir, and K. R. Varadarajan. Approximate shortest paths on a convex polytope in three dimensions. *J. ACM*, 44:567-584, 1997.