The VLDB Journal (2008) 17:1101–1119
DOI 10.1007/s00778-007-0053-2

REGULAR PAPER

A multi-resolution surface distance model for *k*-NN query processing

Ke Deng · Xiaofang Zhou · Heng Tao Shen · Qing Liu · Kai Xu · Xuemin Lin

Received: 22 February 2006 / Revised: 15 October 2006 / Accepted: 16 April 2007 / Published online: 26 June 2007

Abstract A spatial k-NN query returns k nearest points in a point dataset to a given query point. To measure the distance between two points, most of the literature focuses on the Euclidean distance or the network distance. For many applications, such as wildlife movement, it is necessary to consider the surface distance, which is computed from the shortest path along a terrain surface. In this paper, we investigate the problem of efficient surface k-NN (sk-NN) query processing. This is an important yet highly challenging problem because the underlying environment data can be very large and the computational cost of finding the shortest path on a surface can be very high. To minimize the amount of surface data to be used and the cost of surface distance computation, a multi-resolution surface distance model is proposed in this paper to take advantage of monotonic distance changes when the distances are computed at different resolution levels. Based on this innovative model, sk-NN queries can be processed efficiently by accessing and processing surface data at a just-enough resolution level within a just-enough search region. Our extensive performance evaluations using real world datasets confirm the efficiency of our proposed model.

K. Deng (✉) · X. Zhou · H. T. Shen · Q. Liu
School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, QLD 4072, Australia
e-mail: dengke@itee.uq.edu.au

X. Zhou
e-mail: zxf@itee.uq.edu.au

H. T. Shen
e-mail: shenht@itee.uq.edu.au

Q. Liu
e-mail: qing@itee.uq.edu.au

K. Xu
National ICT Australia, Australian Technology Park,
Eveleigh, NSW 1430, Australia
e-mail: kai.xu@nicta.com.au

X. Lin
School of Computer Science and Engineering,
University of New South Wales, NICTA, Sydney,
NSW 2052, Australia
e-mail: lxue@cse.unsw.edu.au

1 Introduction

Efficient processing of k-NN queries in large spatial databases has been investigated extensively in the past. Given a point data set $\mathcal{D}$, a distance function d and a query point q, a k-NN query finds $S \subseteq \mathcal{D}$ such that $|S| = k$ and for any point $p \in S$ and $p' \in \mathcal{D} - S$, $d(q, p) \leq d(q, p')$. Different distance functions can be used. Most of the spatial database literature focuses on the Euclidean distance [18,19]. The network distance has been considered for the applications where object movement is constrained to a pre-defined underlying network, such as a road network or drainage system [14,17]. In this paper, we consider the applications where objects can only move on the surface. For example, an environment scientist to identify animal groupings often uses k-NN queries to study animal inhabitation areas (shapes and positions), their relationships with the environment (their nearest foraging and water sources and human settlement activities) and migration trends (speed and distances). For such studies in mountain areas, the surface distance is a more accurate and realistic proximity measure. Other examples include rover path planning [23], military and utility planning, emergency scene coordination (e.g., to fight bush fires) and low-altitude flight simulation. This new type of k-NN queries using the surface distance is called surface k-NN query (denoted as sk-NN). Note that the surface distance between two points is computed from the shortest path between two points on the surface (see Fig. 1 for example).

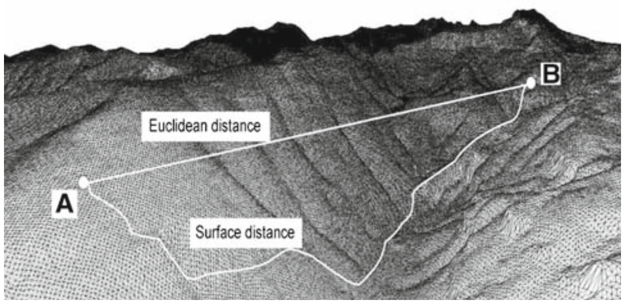


Fig. 1 An example of surface distance

A variety of techniques have been developed for efficiently processing k-NN queries. The existing techniques could be divided into two categories: (1) *constraint-free* k-NN queries and (2) *constraint-based* k-NN queries. For a constraint-free k-NN query, the distance between two points is calculated only based on the coordinates of the two points. Therefore, techniques for processing this type of query focus on minimizing the number of points to be fetched and checked from $\mathcal{D}$. In other words, I/O cost caused by accessing data in $\mathcal{D}$ is a major issue to be considered. In the case of high dimensional space or with a complex distance function, CPU cost should also be minimized.

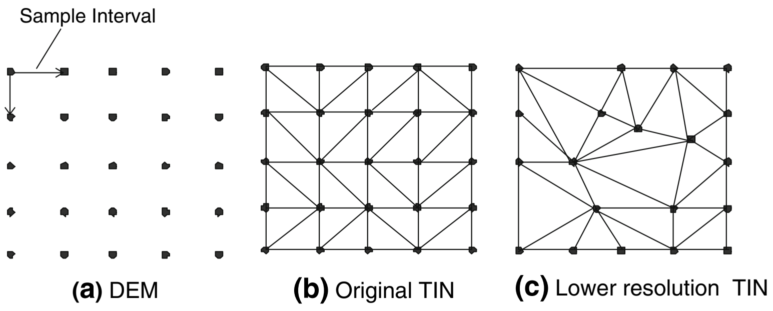
For a constraint-based k-NN query, the distance between two points is calculated not only from their coordinates but also based on the underlying environment data. For example, to find the nearest water source (gas station) for a group of animals (a car) in mountain area (road network), we have to find the shortest path along the surface (network) which leads the animals (car) from the current position to the position of water source (gas station). Here the key efficiency issues to be considered are not only the I/O cost caused by accessing data in $\mathcal{D}$ but also the I/O cost caused by fetching the underlying environment data. The size of environment data could be several orders of magnitude larger than that of $\mathcal{D}$. The distance calculation, which is a frequent operation for any k-NN query processing, also incurs a high computational cost. Therefore, the cost of finding the shortest path in a constrained space is intrinsically high in terms of both I/O cost and CPU cost.

A surface can be represented by a series of discrete digital elevation samples. The U.S. Geological Survey (USGS) Digital Elevation Model (DEM) data files, for example, consist of a sampled array of elevation measurement for a number of ground positions at regularly spaced intervals. We call these sampled data as terrain data. Based on these samples, a surface mesh can be constructed using various data structures. The most popular one is Triangular Irregular Network (TIN). Figure 2a shows an example of the sampled data where each point represents the elevation of the sampled position. Figure 2b shows its corresponding terrain model at its original resolution represented by TIN. In most cases, the number of original sampled data can be very large. To reduce the storage space and improve online visualization performance, a lower resolution terrain model can be produced from a simplification process (see Fig. 2c). In this paper, we use the term of terrain model and terrain mesh interchangeably.

There are three main challenges to efficient sk-NN query processing.

- A digital surface model can consist of millions of terrain data points even for an area of moderate size. For example, in Fig. 1, within a region of $10\,km \times 10\,km$ using $10\,m$ sampling interval, there are about 1.3 million points. Clearly, a k-NN query processing method that only considers organizing the spatial objects to optimize the I/O cost will not be efficient for sk-NN queries because they do not give a careful consideration to the underlying terrain data.
- Traditional pruning techniques for k-NN query processing can be inefficient for sk-NN queries. Typically, the Euclidean distance is used as a lower bound to prune the search space. We found that the ratio of the surface distance over the Euclidean distance between the same pair of points can vary from 200 to 300% for rugged mountain areas, to just 20–40% for some other areas. This could lead to using an unnecessarily large search area for some cases, or repeated search area enlargement (and shortest path calculations) for others.

Fig. 2 An example of terrain model



– The cost of computing the surface distance is extremely high. For some moderately large areas (a few square kilometers), the most efficient surface shortest distance algorithms [1] can take tens of minutes on a modern PC machine to find the shortest distance between one pair of points on the surface, and one of the most efficient approximate surface shortest distance algorithms [11] still takes several minutes. Note that distance calculation is a fundamental operation that is frequently used in any sk-NN query processing.

In this paper, we investigate the problem of efficient sk-NN query process. We approach the problem from two angles: (a) building a multi-resolution terrain model such that the estimated distances using lower resolution data can be used as a guide to restrict the search region where higher resolution data is needed (for more accurate distance estimation); (b) developing fast algorithms for distance ranking by considering lower and upper bounds instead of using accurate surface distances that are costly to compute. The combined advantage is that a sk-NN query can be processed by accessing and processing the data at a *just-enough* level of details (LOD) from a *just-enough* region of interest (ROI), often without computing surface shortest paths. To facilitate these, we propose a multi-resolution surface distance model comprising two data structures: Distance multi-resolution mesh (DMTM) and multi-resolution support distance network (MSDN). DMTM is a multi-resolution terrain model with distance information. It can be used to derive a terrain model at various resolution levels from a lower-than-original resolution level (for those applications that do not need high resolution levels) to a higher-than-original resolution level (for surface shortest distance calculation, as explained later in this paper). It can also be used to estimate the upper bound of the shortest distance at a particular resolution. On the other hand, MSDN contains a set of support distance networks (SDN) at different resolution levels. An SDN consists of a set of selected points from the original surface model, and is used to estimate the lower bound of the shortest distance at a given resolution.

Based on DMTM and MSDN, a novel sk-NN query processing algorithm called multi-resolution range ranking (MR3) is proposed using the filter-and-refine optimization strategy. Our experiments using real terrain data show that MR3 outperforms the benchmark algorithm by up to an order of magnitude in all cases.

A preliminary version of this paper appeared in [3]. There we stressed the multi-resolution data structures and algorithm MR3. Here in this paper, we extend the work with more analysis for the related work and a comprehensive performance analysis (including a new cost model) for our methods to provide significant insights into the behavior and the superiority of our approach. The algorithm reported in the preliminary version requires to fetch and use all terrain data (within an identified ellipse area) for network distance calculation. The step is now further optimized by integrating a connectivity-encoding scheme into the traditional disk-based storage scheme for the multi-resolution data structures to further reduce the I/O overhead. We also provide a new resolution selection scheme that concerns how to decide the initial resolution and the steps for a series of intermediate resolutions during query processing. New experiments are conducted to justify these new features.

The remainder of this paper is organized as follows. After a review of related work in Sect. 2, we present our multi-resolution surface distance model in Sect. 3. Section 4 describes our query processing algorithm, which are analyzed in Sect. 5. A comprehensive empirical performance study is reported in Sect. 6. We conclude this paper and briefly discuss future directions in Sect. 7.

2 Related work

In this section, we will briefly introduce the related work in the areas of k-NN query processing, surface distance computing and multi-resolution terrain modeling.

2.1 k-NN query processing

k-NN query processing has been extensively investigated in Euclidean space and spatial networks [7,14,18,20], and in high dimensional space (e.g., for content-based information retrieval where an object is transformed into a feature vector and object similarity is measured using a distance function) [9,25]. Many variations of the basic k-NN problems have also been investigated.

Constraint-free k-NN query processing focuses on minimizing the number of object data accessed. Typically, a hierarchical spatial index (such as the R-tree) is used to prune the search space by either depth-first or best-first traversal. The former one only visits the index entries with distance smaller than the current kth NN [18] while the latter one only visits entries with the smallest distance of all visited [7]. In a high dimensional space, the VA-file based k-NN query processing [25] proposes to use a distance ranking method. This method uses approximation data to estimate lower bound (lb) and upper bound (ub) of the distances for all objects to the query point q. Let the $(k+1)$ nearest neighbors of q be $\{p_1, p_2, \ldots, p_k, p_{k+1}\}$. The search terminates if $ub(p_k) \leq lb(p_{k+1})$. Otherwise, a refinement using more object data is required to improve the accuracy of the distance estimation.

One type of constrained k-NN query, network k-NN query, has received some attention recently [14,17,20]. In [14], a network Voronoi diagram for objects is pre-computed to

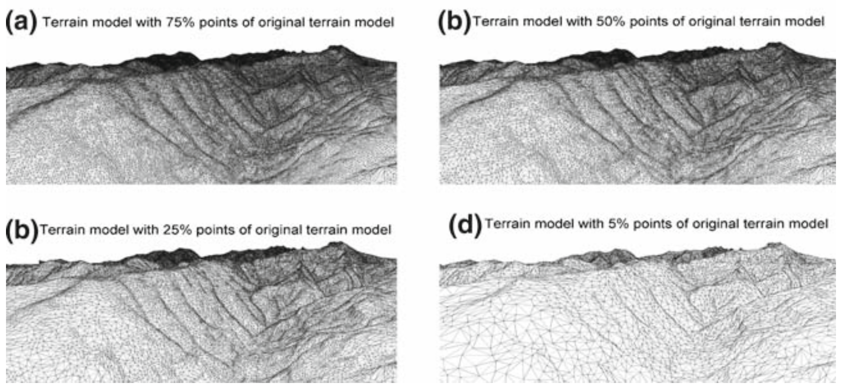

(a) Terrain model with 75% points of original terrain model
(b) Terrain model with 50% points of original terrain model
(b) Terrain model with 25% points of original terrain model
(d) Terrain model with 5% points of original terrain model

Fig. 3 Terrain model with different resolutions

facilitate the online query processing. However, the cost of updating a Voronoi diagram can be high if the network condition changes frequently (e.g., when dealing with moving objects such as vehicles and animals). In the approach proposed by [17], a Euclidean distance based k-NN query is performed first, and the network distance from each Euclidean k-NN point to the query point is then computed. The maximum network distance found during this process is used as a threshold T. If any object p has Euclidean distance to the query point shorter than T, p is deemed as a candidate of the final solution, and its network distance to the query point will be computed to remove false hits. Clearly, the performance of this approach largely depends on whether the Euclidean distance can accurately indicate the network distance. Another approach in [17] gradually expends the search region on the network from the query point. The object reached earlier is closer to the query point. [20] proposes a method that can return an approximate answer to meet the application requirements.

Although a terrain model can be considered as a network, the existing network k-NN techniques cannot process sk-NN queries since the surface distance is not network distance. Moreover, terrain data is much larger and more complex than road networks. For example, while the main road networks of entire North America contain around 0.179 million edges (road segments) (www.maproom.psu.edu/dcw), a small terrain mesh ($10\,km \times 10\,km$ with 10 m elevation sample interval) may contain more than 2.6 million edges (1.3 million points). Therefore, it is very hard (or impossible) to directly process a k-NN query on the original terrain model. To address the challenge caused by the huge terrain data, our solution is inspired by using low-cost distance estimation to restrict the search area [17] and ranking objects without finding the exact surface distances.

2.2 Multi-resolution terrain model (MTM)

A high resolution terrain model consists of millions of points. As not all applications require the same LOD, multi-resolution terrain modelling has been used to dynamically reduce the number of points so that the terrain can be approximated with maximum similarity at a lower resolution tailored for an application, such as realtime display. The work in [5] provides a comprehensive introduction to the multi-resolution terrain modelling techniques. Figure 3 illustrates four lower resolution approximations of the original terrain model shown in Fig. 1. From (a) to (d), the simplified terrain models contain 75 to 5% points of the original terrain model. Note that the major geographical characters of the terrain are maintained even at a very lower resolution. One of the main challenges of multi-resolution database indexing is to efficiently retrieve terrain data with just-enough LOD from a just-enough ROI according to application requirements.

Progressive meshes (PM) [8] is a popular multi-resolution terrain model, where the data at any required LOD and ROI can be derived on-the-fly from database by following a tree structure progressively. Direct mesh (DM) [26] improves the query processing performance of PM by allowing partially materialized surface information using a low-overhead connectivity-encoding scheme. As a result, DM does not need to fetch all internal nodes from the root of the mesh tree in order to obtain the connectivity information. The existing multi-resolution terrain models, however, are designed only for the visualization purpose and contain no distance information. They are not capable to support sk-NN query processing.

2.3 Surface distances computing

Computing the shortest path on a polyhedral surface represented by a triangular mesh is a well-studied problem. The Chen and Han's algorithm [1] computes exact surface distance in $O(n^2)$ time, where n is the number of points in the terrain model. The idea of this algorithm is to unfold all the faces of the polyhedral surface model to the same plane. Figure 4 shows an example to compute the surface distance between point a and b. The triangular face $\triangle cbd$ is unfolded to the same plane as the triangular face $\triangle acd$.

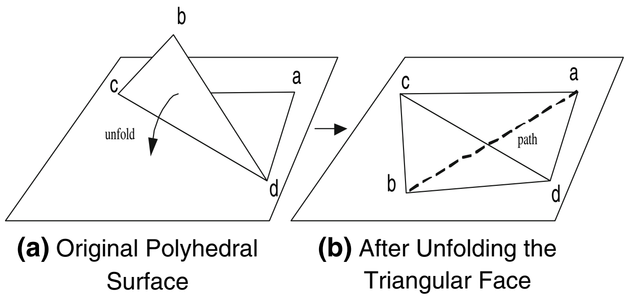

b
c
a
unfold
d
(a) Original Polyhedral Surface
c
a
path
b
d
(b) After Unfolding the Triangular Face

Fig. 4 An example of polyhedral surface unfolding

The surface shortest path is the straight line between a and b as illustrated in Fig. 4b. During the unfolding process, a sequence tree structure is built in memory. Each node of the tree represents a set of shortest paths which all have the same edge sequence and angularly continuous at the query point. The very large size of the tree structure means that the Chen and Han's algorithm is not feasible for large and high-resolution terrain surfaces. In addition, it is not easy to extend this algorithm to use multi-resolution terrain models because the surface distance between the same pair of objects is not same on the terrain models of different resolution levels and cannot change in a monotonic manner (i.e., the exact distance on a lower resolution terrain model is not *always* longer or shorter than the exact distance between the same pair of points on a higher resolution terrain model). In [16], the Chen and Han's algorithm is implemented and the performance is discussed. Its running time is improved by Kapoor [13] to $O(n \log^2 n)$, but the improved algorithm is too complex to be implemented [12].

There are several approximate algorithms for surface distance computing [11,24] where the surface distance is approximated by a computational efficient network distance. The Kanai and Suzuki algorithm is popular due to its simplicity and efficiency [11]. A so-called *pathnet*, which is created by inserting Steiner points into the original surface model, is used. The network in Fig. 5a is a part of the original terrain model, and Steiner points split the edges (shown in Fig. 5b). The links among these points and the original vertices in the same triangular face create new edges in the original surface model (shown as the dashed lines in Fig. 5b). For two given vertices, the network shortest path search operation is performed repeatedly on the *pathnet* with increasing level of resolutions (i.e., more Steiner points are inserted and more connections are available) in a selectively refined region until reaching the required accuracy. This algorithm enjoys a high level of accuracy in practice.

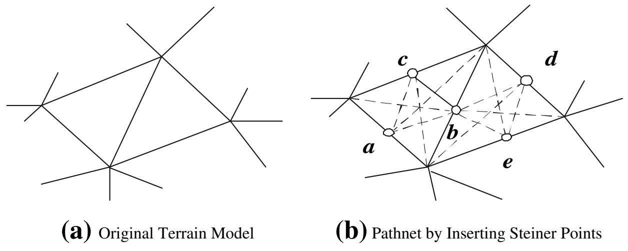

c
d
a
b
e
(a) Original Terrain Model
(b) Pathnet by Inserting Steiner Points

Fig. 5 An example of pathnet

A recent study [2] proposes an accurate surface shortest path algorithm by introducing an early termination condition and a method to select the initial search area coupled with a network expansion strategy, such that the problem of finding the shortest path between two points can be completed in a local region, instead of checking the entire surface as other algorithms do.

3 Multi-resolution surface distance model

The multi-resolution surface distance model is proposed to estimate the surface distance by a distance range in order to minimize the surface data access and avoid expensive exact surface distance computation. Base on this model, the distance can be estimated more accurately on higher resolution data. The multi-resolution surface distance model consists of two data structures, distance multi-resolution mesh (DMTM) and multi-resolution support distance network (MSDN), to compute the distance upper bound (ub) and lower bound (lb).

In this section, we first introduce the ranking scheme and then two data structures of this model are presented.

3.1 Ranking by distance range

The key idea of our approach is to use an MTM such that the surface distance (d_S) can be estimated efficiently at an as low as possible LOD. For any two points a and q on a terrain surface, $d_S(q, a)$ is estimated at resolution r by $lb_r(q, a)$ and $ub_r(q, a)$ such that the condition $lb_r(q, a) \leq d_S(q, a) \leq ub_r(q, a)$ is held. Therefore, $d_S(q, a)$ can be represented by a distance range

$$d_S(q, a) = [lb_r(q, a), ub_r(q, a)].$$

Obviously, the smaller the difference between two bounds is, the more accurate the $d_S(q, a)$ estimation is.

sk-NN query processing can sometimes be done by only using the distance ranges, without using computationally very expensive surface shortest distance algorithms. In Fig. 6, point a, b, c and d are candidates and each line segment is the corresponding distance range to the query point q:

$$d_S(q, a) = [lb_r(q, a), ub_r(q, a)]$$
$$d_S(q, b) = [lb_r(q, b), ub_r(q, b)]$$
$$\vdots$$

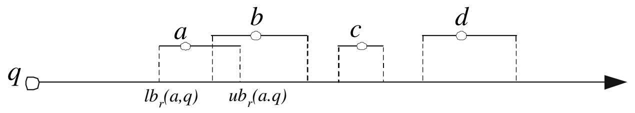


Fig. 6 Distance ranges

To rank any two objects g and h by the distance range, the rules are:

$$\begin{cases} \text{if} \quad [lb_r(q,g), ub_r(q,g)] \cap [lb_r(q,h), ub_r(q,h)] = \varnothing \\ \qquad d_S(q,h) < d_S(q,g), \ \text{if } lb_r(q,g) > ub_r(q,h) \\ \qquad d_S(q,h) > d_S(q,g), \ \text{if } ub_r(q,g) < lb_r(q,h) \\ \text{else} \\ \qquad \textit{cannot rank}. \end{cases}$$

In the example of Fig. 6, c (d) can be ranked as the 3rd (4th) nearest neighbor of q. However we cannot rank a and b since $[lb_r(q,a), ub_r(q,a)] \cap [lb_r(q,b), ub_r(q,b)] \neq \varnothing$. Therefore, such distance range estimations are sufficient for answering a k-NN query from q where $k \geq 2$ but not sufficient for $k = 1$ query, as the distance ranges of a and b are overlapped. In this case, the actual distance $d_S(q,a)$ and $d_S(q,b)$ may need to be computed, using the highest LOD data for accurate surface distance computing (this is often called the *refinement* step, referring the previous step of distance estimation as the *filtering* step which can typically be done very efficiently).

We argue that, using an MTM, it is often sufficient to refine distance ranges on the next level of LOD instead of the highest LOD. To support this, an MTM must support fast distance range estimation, as well as allow progressive improvement of the accuracy of estimated distance ranges when data with a higher LOD is used. Although some network-based approximate surface shortest distance algorithms (e.g., [11,24]) can be used to estimate the upper bound and the Euclidean distance (in either 2D or 3D) can always be used as a lower bound, they do not satisfy the property of continuous improvement. The distance computed by the network-based algorithms on MTM may not monotonically decrease when higher LOD data is used; and the Euclidean distance does not change with LOD.

Next, we introduce the data structure DMTM and MSDN which support fast and monotonic distance range estimation.

3.2 Distance multi-resolution mesh

DMTM is a multi-resolution data structure from which an approximate terrain model can be constructed at variable resolutions. Essentially, it contains a *distance direct mesh* (DDM) which is proposed in this paper on the basis of direct improves direct mesh (DM) [26] by selectively recording distance information, plus a *pathnet* which is obtained by inserting Steiner points into the original terrain model (as in [11]). DDM and the *pathnet* are unified into a single tree structure where those nodes from the DDM are of lower-than-original LOD and used to support progressive upper bound estimation; and those from the *pathnet* are of higher-than-original LOD and used to support approximate surface distance computing.

• Distance direct mesh (DDM)

DDM is based on DM by introducing distance information. It follows the DM's construction process and connectivity encoding scheme to efficiently derive an approximate terrain model of any ROI and LOD. We give a brief introduction of DM below to make this paper self-contained. To build a DM from an original triangular mesh M^n, a sequence of simplifications ($ecol_{n-1}, ecol_{n-2}, \ldots, ecol_0$) are applied.

$$M^n \xrightarrow{ecol_{n-1}} M^{n-1} \xrightarrow{ecol_{n-2}}, \ldots \xrightarrow{ecol_0} M^0.$$

where $ecol$ refers to the operation "edge collapse". Each simplification step produces a simpler mesh, i.e. M^{i-1} (results from $M^i \xrightarrow{ecol} M^{i-1}$) has fewer vertices than M^i. Thus, the sequence of meshes have a monotonically reducing number of vertices. The inverse of the simplification is the refinement.

$$M^0 \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1}, \ldots \xrightarrow{vsplit_{n-1}} M^n.$$

where $vsplit$ refers to the operation "vertex split". By applying a sequence of refinements, the mesh of any resolution level (including original one) can be rebuilt from the simplest mesh M^0. In DM, the simplest mesh M^0 and refinement $R = \{vsplit_0, vsplit_1, \ldots, vsplit_{n-1}\}$ form the multi-resolution representation of the original mesh M^n. In Fig. 7, the simplification step $M^7 \xrightarrow{ecol} M^6$ is by collapsing edge v_1v_2 and refinement step $M^7 \xleftarrow{vsplit} M^6$ is by splitting vertex v_9.

DM is organized by a binary tree (see Fig. 7b). All the leaf nodes form the original terrain mesh, and each non-leaf node represents a lower resolution approximation of its descendants. DM construction is a bottom-up process. Each vertex in the original terrain mesh is represented by a leaf node. Then, a pair of connected nodes are selected to collapse to form their parent node if the resultant terrain after the collapse causes minimum approximation error according to some error measure (e.g., the quadric error matrices [6]). Such approximation error e is recorded by every non-leaf node. This process continues until a tree is formed (so the entire terrain is approximated by one point). Figure 7a shows the process to simplify a mesh M_7 to M_6 by collapsing edge v_1v_2 and this process is recorded in the DM tree structure (Fig. 7b). Through a series of similar edge collapses, the mesh M_7 is eventually simplified to a single point v_{15}. In this paper, a higher LOD means a high resolution and smaller approximation error.

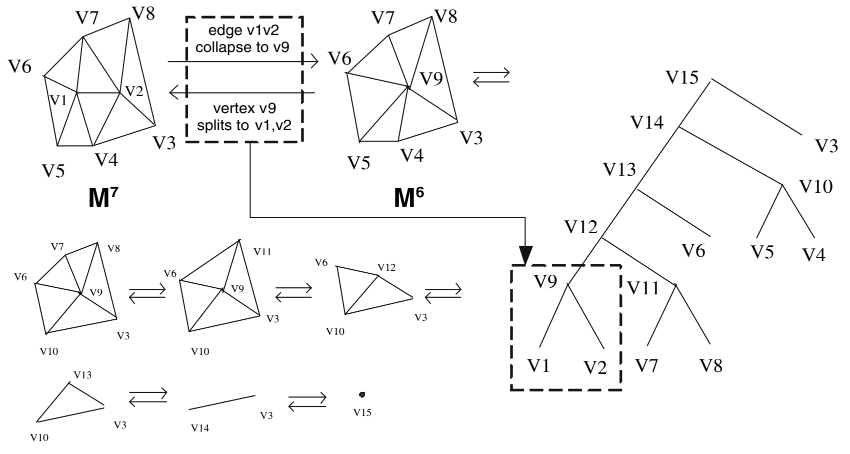

edge v1v2 collapse to v9
vertex v9 splits to v1,v2
M⁷
M⁶
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15

Fig. 7 An example of terrain mesh simplification

(a) Terrain Mesh Simplification **(b)** Tree Recording the Simplification Process

The process of reconstructing a terrain model for a given LOD and ROI is a top-down process. It starts from the root and expands by following the tree until the required LOD and ROI conditions are met. DM implements a compact connectivity-encoding scheme, to let each node v record a list of identities (IDs) of the nodes to which v may connect and they have a *similar* LOD. Two nodes are said to have a similar LOD if their LOD *intervals* overlap, where the LOD interval of node v is $[v.children.e, v.parent.e)$ (the approximation errors of v's children and parent node). This connectivity encoding scheme used in DM abolishes the need of level-by-level tree expansion starting from the root of an MTM tree (in order to obtain connectivity information among the nodes), and the concept of similar LOD is used to limit the number of nodes to which a node needs to record connectivity information.

Now, we introduce our approach to add the distance information to built DDM. Note that DM is designed for terrain visualization only. It does not support (1) fast distance estimation, (2) monotonic change when distances are computed at increasing (or decreasing) LOD. DDM can support both of them, by adding distance information to each edge based on the same DM connectivity-encoding scheme. A distance is recorded during DM tree construction. If two leaf nodes a and b are connected (i.e., from the original mesh), the length of the edge in the original mesh is the distance and is recorded in both node a and b. Each node's *representative node* in the original mesh, is itself. Every node in DDM has a representative node in the original mesh. The importance of this property will be discussed later. Let $N(v)$ represent the neighbors of node v. When the edge linking node a and b is selected by the DM construction algorithm to collapse into c, $N(c) = N(a) \cup N(b)$ (same as the DM construction algorithm). The representative node of c is set to be the representative node of either a or b (say, a). For each node $w \in N(c)$, $d(c, w)$ is defined as:

$$d(c,w) = \begin{cases} d(a,w) & \text{if } w \in N(a) \\ d(b,w) + d(a,b) & \text{if } w \in N(b) - N(a). \end{cases}$$

• *Pathnet* The second part of DMTM is a *pathnet*, which is created by inserting Steiner points into the edges of the original terrain model as shown in Fig. 5. *Pathnet* provides passageways crossing the inside of the triangular face, which are not traversable before. If more Steiner points are inserted, the network distance can approach the exact surface distance to a very high level of accuracy.

• *Property of DMTM* Assume that both DM and a *pathnet* are necessary to support multi-resolution terrain applications, the extra storage overhead of DMTM is very small (just adding a distance value to each stored edge in DM).

Next, we discuss the property of DMTM which supports monotonically improvable upper bound computation (i.e., upper bound decreases when computing on higher resolution data). A terrain mesh of a given LOD and ROI can be derived from DDM, just as in DM (or directly from *pathnet*). A derived terrain model is a network. Therefore, Dijkstra's shortest path algorithm [4] can be used to compute the network distance between a pair of object points. If an object point is not on a vertex of the derived terrain model, it needs to be transformed to be a network vertex in order to compute the network distance. On the original terrain model or *pathnet*, if an object p locates on a triangular face (called local face), say $\triangle ABC$, we can simply connect p to each end point of $\triangle ABC$ using a straight line and the Euclidean distance between them is recorded. For simplicity, in this paper, we suppose all objects are located on the network vertices of the original terrain model. If the terrain model resolution is lower than the original level, the transformation of p identifies the local triangular face on this simplified model, say $\triangle DEF$, and p is linked to each end point of $\triangle DEF$.

Different from the case of original terrain model and *pathnet*, the distance between p and each end point is the network distance between their representatives on the original terrain model. Next, we prove that such network distance is the upper bound of their surface distance.

Property 1 *The network distance between two object points a and b on the terrain model of LOD level r derived from DMTM is the upper bound of their surface distance on the original terrain model.*

Proof For any two vertices a and b on the derived terrain model from DDM, the network shortest path between them is a sequence of edges. Note that the length of each edge in this model is the distance of a path on original terrain model from the representative of one end (of the edge) to the representative of another end (of the edge). Therefore, the shortest path between a and b corresponds to an actual network path on the original terrain model from the representative of a to the representative of b. It must be longer than or equal to the surface distance between a and b on the original terrain model, i.e. the upper bound of the surface distance.

On the *pathnet*, it is obvious any network distance between a and b is longer than their surface distance. □

The following property proves that the upper bound monotonically decreases when computing on higher resolution data.

Property 2 *For two object points a and b on the terrain model of LOD level r derived from DMTM, let the upper bound be denoted as $ub_r(a, b)$. It can be guaranteed that for LOD level $r' > r > r''$, $ub_{r'}(a, b) \le ub_r(a, b) \le ub_{r''}(a, b)$ is held.*

Proof The higher resolution terrain model has more extra edges based on the lower resolution terrain model (i.e., all edges of the lower resolution model form a subset of the edges of the higher resolution model). Since there are more path choices, the shortest path from a to b found on higher resolution terrain model must be shorter than or equal to that found on lower resolution terrain model. That is, the upper bound decreases along with the increase of LOD.

When the resolution level increases over the original terrain model resolution (*pathnet*), this property can be proved in the similar way. □

Figure 8 shows an example of Properties 1 and 2. The length of edge ab is the distance of the path $a'b'$ on the original terrain model, so do the other edges. Suppose the network shortest path from a to b on the terrain model in (a) is the edge ab. In the higher resolution terrain model (b), more edges are added and the network shortest path from a to b may be changed if $ad + db < ab$ where $ad + db$ corresponds to $a'd' + d'b'$ which is on original terrain model. Therefore, the upper bound of the surface distance between a and b increases when computing on higher resolution data.

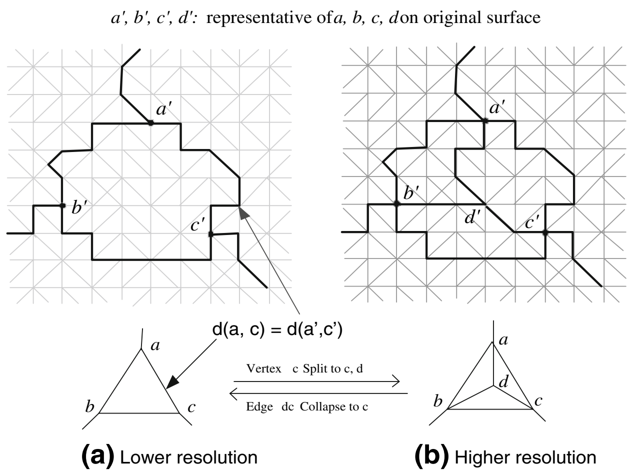


Fig. 8 An example of upper bound finding on different resolution terrain models

3.3 Multi-resolution support distance network

MSDN, inspired by the plane-sweep algorithm, is designed to support fast and progressively improvable lower bound estimation (i.e., lower bound increases when computing on higher resolution data). It consists of a set of support distance networks (SDN) at different resolutions.

We explain the intuition behind SDN first. While $d_E(a, b)$ can be used as a safe lower bound of $d_S(a, b)$, it is not tight and its accuracy cannot be improved by using higher LOD environment data. Consider a terrain in 3D space, where the z-axis represents the height. Assume $a.y < b.y$. By using a 2D plane $y = y_0$ $(a.y < y_0 < b.y)$ to cut through the terrain, a polyline l (called a *crossing line*) can be obtained by intersecting the plane with the terrain surface. Then, any surface path from a to b must pass l at least once. For a point p on l, if $d_E(a, p) + d_E(p, b) \le d_E(a, p') + d_E(p', b)$ for any other point p' on l, then $d_E(a, p) + d_E(p, b)$ is a better lower bound of $d_S(a, b)$ since it is much tighter than $d_E(a, b)$. The accuracy of lower bound estimated in this way can be improved when more y-planes are used. Clearly, a y-plane is not useful if $a.y = b.y$; and in this case, x-planes should be used. To cater for arbitrarily positioned points, both x- and y-planes need to be prepared, and the angle between the projection of (a, b) on the (x,y)-plane and the x-axis is used as a heuristics to choose which set of planes to be used. If the angle is less than 45°, a set of y-planes will be used; x-planes otherwise. Figure 9 is an example where one 2D plane $y = y_0$ cuts through the original terrain model. On the crossing line, p is the point from which the summation of the Euclidean distances to a and b is minimum. Therefore, $d_E(a, p) + d_E(p, b)$ can be used as the lower bound of the surface distance of a and b.

Denote the crossing line obtained by intersecting the terrain (at the original resolution) and plane $y = y_0$ as l_{y_0}. l_{y_0} is

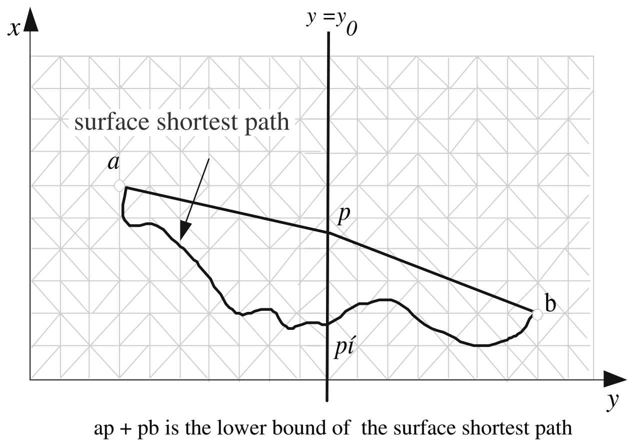


Fig. 9 Lower bound finding by using one 2D plane $y = y_0$

a polyline with a sequence of points $\{(x, y_0, z)\}$. We define its resolution as 100%. A polyline can be approximated by fewer points using some line simplification algorithms, such as [15] which can reduce the number of points while maintaining a maximum level of similarity between the lines before and after simplification. However, such an algorithm needs to be modified to ensure that MSDN can be used to estimate the lower bound with monotonic increase of accuracy with higher resolution data. We consider two consecutive points in a crossing line (a polyline) as an MBR. Our modification is to ensure that the MBR of the simplified line segment must fully enclose the MBRs of every line segment from the line segment before simplification. If l'_{y_0} is an approximation of l_{y_0} using $r\%$ points of l_{y_0}, we say the resolution of l'_{y_0} is r. By placing a set of x- and y-planes in the space, the set of crossing lines obtained form a SDN (with 100% resolution). An SDN at resolution level r is obtained from simplifying every crossing line in the 100% resolution SDN by using $r\%$ points for each crossing line. MSDN is then defined as a collection of SDNs with different resolutions.

Using a SDN at resolution r to estimate $lb_r(a, b)$ needs to use Dijkstra's network shortest distance algorithm. A network is constructed from the SDN by treating each line segment as a node. For each node, there is an edge to link each of the nodes which are line segments from the neighboring crossing lines. The length of an edge is the minimum Euclidean distance between the MBRs of the two line segments. Point a and b also need to be transformed into the network by connecting them with the nodes from the first crossing line on the plane they encounter when moving one point to another along a straight line. Note that only the SDN in a restricted area is required for lower bound computation for two given points (see the next section for detailed discussions), and not all planes need to be used for low resolution estimation. Therefore, it is unnecessary to materialize the connection information for the entire SDN, which can be very large; they are computed on-the-fly when they are retrieved for lower bound estimation. It is easy to see that, when more planes are used, or higher resolution SDN is used, such an estimated lb is getting longer and further approaching the shortest surface distance. In summary, SDN has the following properties.

Property 3 *The network distance between two object points a and b on a SDN network at resolution level r is the lower bound of their surface distance on the original terrain model.*

Proof For any two objects a and b on the terrain model, the shortest network path $d_N(a, b)$ between them in SDN must pass through a series of crossing lines. The surface shortest path, corresponding $d_S(a, b)$, must pass through these crossing lines as well and a path $path$ can be created by connecting a through these passing points to b using straight line segments. It is clear that $path \leq d_S(a, b)$. Note that the $path$ is actually a path in SDN. Its distance must be longer than or equal to the network shortest path between a and b in SDN, $d_N(a, b) \leq path$. Therefore, $d_N(a, b)$ is the lower bound of $d_S(a, b)$. □

In the following property, we prove the lower bound increases when computing on higher resolution data.

Property 4 *For two object points a and b on a SDN network of resolution level r, let the lower bound be denoted as $lb_r(a, b)$. It can be guaranteed that for resolution level $r' > r > r''$, $lb_{r'}(a, b) \geq lb_r(a, b) \geq lb_{r''}(a, b)$ is held.*

Proof For every crossing line in the SDN at lower resolution level, each point is an MBR which approximates several points in the SDN at higher resolution. Since the distance between two MBRs is shorter than or equal to that of any pair of their children points (each from one MBR), the network shortest path between a and b found in the SDN at lower resolution must be longer than or equal to that found in the SDN at higher resolution. That is, the lower bound found in SDN increases when the resolution of SDN increases. □.

The planes used to generate MSDN can be placed strategically according to terrain roughness (i.e., more dense planes for more rugged region). To ensure an estimated distance using MSDN can be as close as to d_S, the planes can be placed at the highest density for some regions with the interval that is equal to the average length of edges in the original terrain mesh.

3.4 Multi-resolution disk storage scheme

The multi-resolution surface distance model is stored separately from the data objects. There are three advantages by doing this. First, the data management is independent from each other. Second, there are normally different types of data collected from the same region for various purposes. One separately stored surface model can uniformly provide topography of this region for different applications. Finally, while a large number of surface approximations of different LODs can be derived from the multi-resolution terrain model, only a few of them are used in query processing. In this situation, it is not wise to map data objects into all resolution level terrain models. In our method, the location of each candidate is found on-the-fly after the terrain model of a required LOD has been derived.

As discussed before, the upper and lower bound of the surface distance are network distances and Dijkstra's algorithm is used for network distance computation due to its efficiency and simplicity. In Dijkstra's algorithm, a "wavefront" expands from the source vertex until the destination vertex is reached. A heap is used to keep the frontier nodes on the "wavefront" together with the distance to the source vertex. In large network, the secondary-memory version of Dijkstra's algorithm is used to just keep frontier nodes in memory to minimize the memory requirement [10,21]. The "wavefront" is expanded by deleting the frontier node with minimum distance to the source vertex in the heap and inserting all adjacent vertices of the deleted one. Therefore, by clustering network vertices based on network connectivity, the adjacent vertices of each frontier node can be retrieved by one (few) disk page access [22,17,27]. In this network storage scheme, the I/O of the Dijkstra's algorithm can be minimized. However, the existing storage schemes only consider single resolution network. They need to be adapted for the multi-resolution surface distance model in order to minimize the memory requirement as well as I/O.

• *Storage scheme for upper bound*
First, we discuss the situation of DMTM, the model for upper bound computation. In traditional network storage scheme, if two vertices are linked by an edge, they are adjacent each other. In case of a single resolution network, all adjacent vertices of each vertex are sequentially stored in the same disk page. If current page is full next page will be used. Let L_v be the adjacent list of a vertex v. Each adjacent vertex $v' \in L_v$ is in the form

$$<v', dist(v, v'), L_{v'}>.$$

where $L_{v'}$ is a pointer referring to the adjacent list of v' and $dist(v, v')$ is the distance of the edge linking v and v'. In DMTM, the situation becomes more complex since, for each vertex, its adjacent vertices belong to different resolution levels. We introduce the connectivity-encoding scheme proposed in Direct Mesh into the traditional storage scheme. The connectivity-encoding scheme decides whether two adjacent nodes are in the same resolution level by recording LOD for each node. Each $v' \in L_v$ is in the form

$$<v', v'.children.e, v'.parent.e, L_{v'}, dist(v, v')>.$$

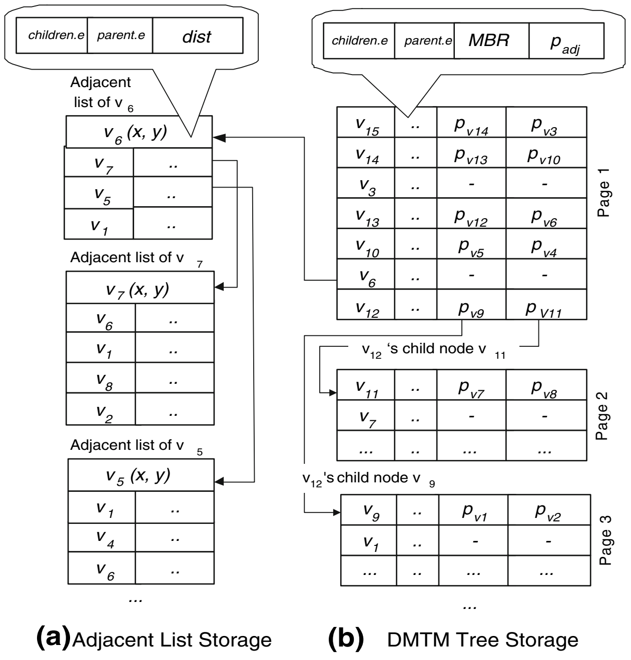


Fig. 10 Storage scheme of DMTM

where $[v'.children.e, v'.parent.e)$ is the LOD interval of v' (see Sect. 3.2). When the upper bound is searched at a resolution r using Dijkstra's algorithm, the frontier node with the minimum distance to the source in the heap is replaced by those of its adjacent nodes with proper LOD interval, i.e. $children.e \leq r < parent.e$. Figure 10 (a) is the adjacent list storage scheme for the multi-resolution network data structure in Fig. 7.

Recall the object needs to be transformed to be a network vertex in order to compute the network distance as described in Sect. 3.2. In the transformation, the local triangular face can be identified by browsing the DMTM tree. In DMTM tree, each node keeps a 2D minimum bound rectangle (MBR) of its adjacent nodes on the (x, y)-plane. Given the candidate p and resolution lever r, the tree traverse starts from root node and only the entries whose MBRs enclose the projection of p on (x, y)-plane are visited until the node of required resolution is reached (i.e. for a node v, $v.children.e \leq r < v.parent.e$). After the local triangular face of p is found, the distance from p to its three end nodes are computed on the original terrain model using Dijkstra's algorithm. Although DMTM tree is not necessary to be balanced, the difference of the heights in the tree (the number of middle nodes between root node and a leaf node) for large terrain model is similar in most cases [26]. Therefore, in average, $\log_2 n$ nodes are accessed to find a network vertex where n is the number of nodes in DMTM.

DMTM tree is clustered on the disk based on the connectivity to minimize the disk page access. For any node v, it is stored in the same page as its parent node; if the page is full, it will be assigned a new disk page. Thus, nodes in DMTM tree having the same parent node are most likely to be stored in the same disk page. In Fig. 10b, the storage scheme of the example in Fig. 7 is shown. The root node of the tree is stored in page 1 and a new page is assigned to v_9 and v_{11} separately. Each node of DMTM tree is in the form

$<v, v.children.e, v.parent.e, MBR_v, p_{c1}, p_{c2}, p_{adj}>$

where p_{c1}, p_{c2} and p_{adj} are pointers referring to v's child nodes and adjacent list. The proposed multi-resolution storage scheme supports following essential functions.

- **rep**(v); v's representative on the original terrain model is returned.
- **transform**(p, r): at resolution level r, the local triangular face of p is identified and the distance from each end point v of this triangular face to p is computed on the original terrain model (Note both p and $rep(v)$ are vertices of the original terrain model).
- $\mathbf{d_N}(s, t, r)$: on the terrain model of resolution level r, the network distance from object s to t is returned. First, s and t are transformed as vertices of the terrain model by invoking function **transform**. Then, s and t are treated as two normal vertices of the terrain model and $d_N(s, t)$ is computed by using secondary-memory version of Dijkstra's algorithm.

• *Storage scheme for lower bound*
Next, we discuss the storage scheme of MSDN for lower bound computation. In MSDN, the resolution levels are discrete and evenly fixed from low level to high level. At each level, the crossing lines are stored on disk pages one by one. For each crossing line, the nodes are stored in an ascendant order of x value for y-planes (or y value for x-planes). Each node v is in the form $<v, MBR, p_{\mathrm{adj}}>$ where p_{adj} refers to v's adjacent list. v's adjacent list is consisted of vertices of neighboring crossing lines. The other nodes on the same crossing line as v have the same adjacent list. As indicated in Sect. 3.3, the lower bound computation between two objects is conducted within a region guided by the upper bound (discussed in details in next section). Therefore, only part of the SDN needs to be accessed. To minimize the I/O, the crossing lines of each SDN are clustered on the disk and referenced by an index-tree (see Fig. 11). By traversing the tree, we can efficiently identify the disk pages containing the crossing lines which are within the search region of SDN at resolution r.

4 *sk*-NN query processing

In this section, we present algorithm multi-resolution range ranking (MR3), an efficient algorithm for sk-NN query processing based on multi-resolution surface distance model. We first give an outline of the algorithm. Then we discuss in details for optimizing search regions and how upper bound (ub) increases and lower bound (lb) decreases by using higher resolution data but in a reduced search region. At the end of this section, the resolution selection scheme is discussed.

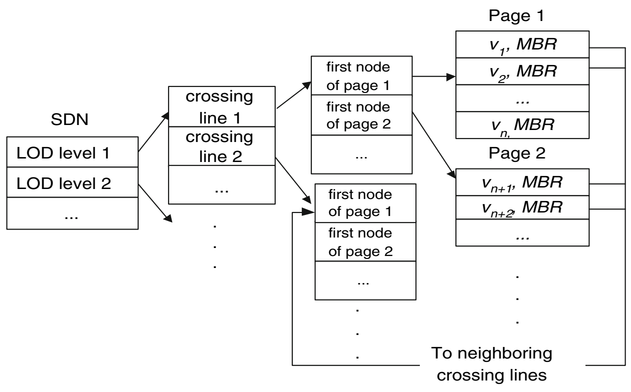


Fig. 11 Index for MSDN

4.1 Algorithm MR3

Given a set of object data $\mathcal{D}$, a terrain surface $\mathcal{S}$, a query point q on the terrain, and an integer k, our task is to find the k nearest neighbors of q on the terrain from $\mathcal{D}$. In order to perform this task, Algorithm MR3 needs to use the following data structures of Multi-resolution Surface Distance Model: (1) a DMTM; and (2) an MSDN (at a pre-determined number of resolutions). Both DMTM and MSDN are derived from $\mathcal{S}$. The major procedure of algorithm MR3 is outlined in Fig. 12.
This algorithm can be sketched in four steps as below. For the sake of description, let $\mathcal{D}^{xy}$ be a set of points which are projections of all points in $\mathcal{D}$ on the (x,y)-plane.

1. *2D k-NN Query* (line 1): Perform a 2D k-NN search in $\mathcal{D}^{xy}$ to find $C1[1..k] \subseteq \mathcal{D}$ whose projections on the (x,y)-plane are the k nearest neighbors to $\overline{q}$.
2. *Surface distance calculation* (line 2–9): The k points in $C1$ will be ranked to find the kth neighbor of q on $\mathcal{S}$, using the algorithm described in the next section, based on multi-resolution surface distance model (DMTM and MSDN). Let this point be b and the estimated distance upper bound is $ub(q, b)$.
3. *2D range query* (line 10–11): A normal range query will be performed on $\mathcal{D}^{xy}$ using $\overline{q}$ as the center and $ub(q, b)$ as the radius, and all the points retrieved are in set $C2 \subseteq \mathcal{D}$.
4. *Surface distance ranking* (line 12–19): All the points in $C2$ will be ranked, using the same algorithm as in step 2, such that the estimated upper bound of the kth neighbor

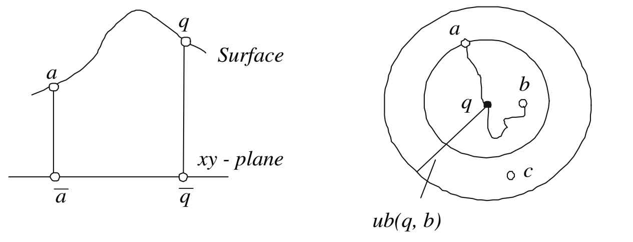
Algorithm MR3

```
Input: Multi-resolution Surface Distance Model: DMTM and MSDN,
       Data object set $D$,
       Query point $q$, the number of nearest neighbors $k$.
Output : $k$ surface nearest neighbors of $q$.
       // $\overline{q}$: projection of $q$ on $(x, y)$-plane
       // $\overline{p}$: projection of $p \in D$ on $(x, y)$-plane
1.     Find a subset $C1$ from $D$, $|C1| = k$, for any $p \in C1$
       and $p' \in D - C1$, $d_E(\overline{p}, \overline{q}) < d_E(\overline{p}', \overline{q})$ is held;
2.     for each object $p \in C1$ do
           //computed on initial resolution terrain
           //model derived from DMTM;
3.         $ub := d_N(p, q)$;
4.         $lb := d_E(p, q)$;
5.     endfor
6.     while object $p$ in $C1$ cannot be ranked by $[lb, ub]$ do
           //on next higher resolution terrain
           //model derived from DMTM
7          $ub := d_N(p, q)$;
8          $lb := d_N(p, q)$; //on next higher resolution SDN
9.     endwhile
10.    $T$ := maximum $ub$ in $C1$;
11.    Find a subset $C2$ from $D$, for any $p \in C2$,
       $d_E(\overline{p}, \overline{q}) < T$ is held;
12.    for each object $p \in C2 - C1$ do
           //computed on initial resolution
           //terrain model derived from DMTM
13.        $ub := d_N(p, q)$;
14.        $lb := d_E(p, q)$;
15.    endfor
16.    while object $p$ in $C2$ cannot be ranked by $[lb, ub]$do
           //on next higher resolution terrain model
           //derived from DMTM
17.        $ub := d_N(p, q)$;
18.        $lb := d_N(p, q)$; //on next higher resolution SDN
19.    endwhile
20.    return first $k$ objects closer to $q$ in ranked $C2$;
```

Fig. 12 MR3 algorithm

of q is not greater than the lower bound of the (k+1)th neighbor of q in $C2$.

The first 3 steps are illustrated in Fig. 13. Note that steps 1 and 3 are 2D spatial queries, which can be processed efficiently. For example, for 2D k-NN query, it can be performed using one of several 2D k-NN query processing methods (e.g., [7,9,18]) if $|\mathcal{D}^{xy}|$ is very large. Note that the first and third step can be done in 3D (i.e., to perform a 3D k-NN query using the Euclidean distance). In the case where the height difference of an object to the query point is very large, it is better to use the Euclidean distance in 3D. The processes for steps 2 and 4 are the same, except that step 2 needs an extra step to calculate an as tight as possible upper bound for the kth neighbor. This distance will be used as the search radius in step 3, which in turn supplies the points that need to be ranked in step 4.



(a) xy-plane Projections **(b)** sk-NN Query Processing (k=2)

Fig. 13 An example of surface k-NN query

The correctness of MR3 is straightforward. Any points not selected in $C2$ must have their Euclidean distance to q longer than $ub(q, b)$, and there are already k points found which have their upper bound distance less than $ub(q, b)$.

4.2 Surface distance ranking

Now we describe the process of surface distance ranking to rank a set of candidate points by their estimated distance ranges, based on the multi-resolution surface distance model. This is used in both steps 2 and 4 in MR3. First, the initial resolution levels of MSDN and DMTM are determined. The lower bound for each candidate point is set to be the Euclidean distance between q and the point. The search region (ROI) for each point is the area from which the environment data needs to be retrieved for distance range estimation. The search region is set to the entire terrain (we will further discuss this in the next two subsections). Then DMTM and MSDN are retrieved according to the values of ROI and LOD. From here, the upper and lower bounds are estimated alternately for each candidate point until the kth neighbor of q can be safely identified. Details of estimating the upper and lower bound using DMTM and MSDN, and the way to embed q, have already been described in Sect. 3. If the kth neighbor cannot be determined by the current set of estimated distance ranges, a higher resolution data (for both DMTM and MSDN) is required, but with potentially few number of remaining candidates (i.e., those points which can be ranked safely as in or out of the final solution set can be dropped), and the search region for them will be reduced again (the details in refining the search region for DMTM and MSDN will be discussed in the next two subsections). The algorithm terminates either when the kth neighbor has been identified, or the highest resolutions of both DMTM and MSDN have been used.

4.2.1 Estimating upper bound

For each candidate, the upper bound estimation starts from the initialized resolution. In order to find the first global optimum upper bound (corresponding to the global optimum

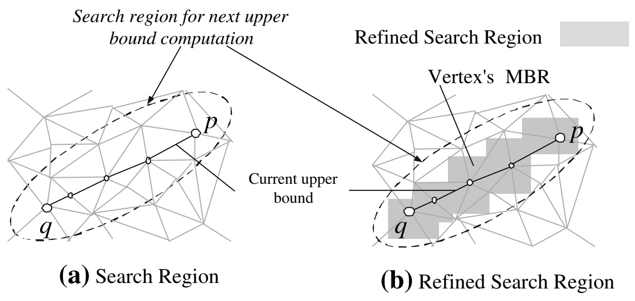

Search region for next upper bound computation
Refined Search Region
Vertex's MBR
Current upper bound
p
q
(a) Search Region
(b) Refined Search Region

Fig. 14 An example of search region refinement

shortest network path) on this resolution, we use the entire DMTM terrain model as the search region.[1] If a candidate cannot be ranked, the upper bound estimation process continues to use the next higher resolution of DMTM data (at a pre-determined interval; see Sect. 5 for the impact of choosing such intervals). The search region will be reduced to the area whose projection on the (x, y)-plane is an ellipse-like area instead of the entire surface (as in [2]). This is shown in Fig. 14a. The ellipse's foci are the projections of q and the candidate point p on the (x,y)-plane. The ellipse's constant is the current estimated upper bound value. Therefore, as the DMTM terrain model resolution increases, the estimated upper bound becomes more accurate (i.e., smaller), and then leads to a reduced search region.

Although the ellipse-like search region is a fraction of the entire surface, it might be still very large considering the Dijkstra's $O(n^2)$ complexity, in particular when the ellipse is approximated by its MBR. We observe that the terrain model with low resolution retains the major geographical characteristics of the original as shown by examples in Fig. 3. Therefore, given two objects, it is more likely that the shortest surface path on the higher resolution terrain model follows the similar track on the lower resolution terrain model. Motivated by this observation, without losing the DMTM's property, the ellipse-like search region can be further pruned to a selectively refined search region. In Fig. 14b, the refined search region is a set of MBRs. Each MBR is consisted of nodes which are the descendants in the DMTM tree of the vertices on the path. The refined search region gradually becomes narrower. If it is too narrow to compute the shortest network path, its area will be expanded by doubling each vertex's MBR. Note that, when using a collection of smaller MBRs instead a large MBR for the ellipse, the CPU and I/O costs can be reduced but the estimated ub may be not as tight as the case when all the data from the ellipse area is used. Nonetheless, any ub estimated in this way remains as a valid upper bound.

[1] The first global optimum upper bound can also be found by using method proposed in [2]. For description clarity, we use the entire surface as the initial search region. Notice that the computation cost is not high, as the initial resolution is very low (e.g. 0.5% of the original one).

4.2.2 Estimating lower bound

As already mentioned, the lower bound is initially set to the Euclidean distance. If an object's rank cannot be identified using the initialized one, the computation starts at a low resolution SDN and iterates at a higher resolution until this object's rank is identified. For each candidate, its ellipse-like upper bound search region can also serve as its lower bound search region. However, for the purpose of lb estimation, the ellipse area cannot be reduced as what we did for estimating ub. Thus, our optimization focus is to reduce the CPU costs (recall that we use Dijkstra's shortest path algorithm to find the shortest path for lb estimation in a SDN). Once a lb is estimated for a candidate point from the lower resolution SDN, the following process will be used to reduce the CPU cost. We propose a concept of *dummy lower bound*, which is estimated using a small part of the ellipse-like search region. This can be done by building an *envelope* which extends the lb path identified from the previous round (i.e., by making it "thicker"), and use those SDN nodes (and edges) that are enclosed by the envelope. The rational is that a lb estimated in this way is greater than or equal to the lb estimated using the entire ellipse-like search region of SDN. Thus, if the distance range using the estimated ub with this lb cannot differentiate this candidate point, a true lb (estimated by using the entire ellipse-like search region SDN) is not possible to differentiate either (as it can only increase the extent of distance range overlapping). But if such the distance range can do that, the true lb needs to be computed in the entire ellipse-like search region on SDN at this resolution level in order to confirm the result of the dummy lower bound. If it is confirmed, lb computation stops. Clearly, the first lb needs to be computed in the entire ellipse-like search region (but with very low resolution SDN).

4.3 Resolution selection scheme

Next, we discuss the resolution scheme which concerns the optimal initial and intermediate resolution selections in the query processing. Table 1 lists the symbols used in the rest of this paper.

First, we discuss the selection criteria of resolution scheme of DMTM for upper bound computation. As described in Sect. 3.4, for two data objects s, t on a terrain model of resolution r (derived from DMTM), the network distance computation includes two steps: (1) the object transformation (2) the shortest path searching. In the first step, function **transform(s,r)** is invoked to compute the network distances on the original terrain model between the representatives of s and each end point of s's local triangular face, so does t.

Table 1 Symbol and definition

Symbol	Definition
λ	Object density: the number of objects whose projections in one unit area on (x,y)-plane
$\mathcal{N}_o$	The number of vertices on the entire original terrain model
$\mathcal{N}_r$	The number of vertices on the entire terrain model at resolution level r
$\mathcal{A}$	Area of entire terrain model
$\|C\|$	Cardinality of dataset C
$\max(d_S(C,q))$	The maximum of $d_S(C[1],q)$,..,$d_S(C[m],q)$, C is the dataset $\{C[1]..C[m]\}$
$d_E^{xy}(s,t)$	The Euclidean distance between the projections of s, t on (x, y)-plane
δ	d_S/d_E^{xy}

Then, function $d_N(s,t)$ computes the network distance on terrain model of resolution r.

For the initial upper bound computation, let n_1 be the network vertices of original terrain model accessed in one object transformation. n_1 can be estimated by:

$$n_1 = \mathcal{S}_1 \cdot \frac{N_o}{\mathcal{A}} = \frac{\mathcal{A}}{\mathcal{N}_r} \cdot \frac{\mathcal{N}_o}{\mathcal{A}} = \frac{\mathcal{N}_o}{\mathcal{N}_r}. \tag{1}$$

where S_1 is the search region in the object transformation and can be estimated by the average area of each triangular face on terrain model of resolution r, i.e. $\frac{\mathcal{A}}{\mathcal{N}_r}$. For all objects in the candidate set C and a query point q, the total network vertices accessed for transformation is $n_1 \cdot (|C|+1)$ since each object as well as q need to be transformed. In the second step of initial upper bound computation, the Dijkstra's algorithm expands the "wavefront" from q until the network shortest paths to all objects in C are reached. Thus, the search region can be approximated by a spherical region and the area is:

$$\mathcal{S}_2 = \pi \cdot \max(d_S(C,q))^2. \tag{2}$$

Then, at the second step, the number of vertices accessed is

$$n_2 = \mathcal{S}_2 \cdot \frac{\mathcal{N}_r}{\mathcal{A}}. \tag{3}$$

The total number of network vertices accessed in two steps for computing initial upper bound is a function of $\mathcal{N}_r$

$$n_{\text{initial}} = n_1 + n_2 = \frac{\mathcal{N}_o}{\mathcal{N}_r} + \frac{\mathcal{S}_2}{\mathcal{A}}\mathcal{N}_r. \tag{4}$$

From this formula, the minimum n_{initial} can be derived by

$$\frac{\mathrm{d}(n_{\text{initial}})}{\mathrm{d}(\mathcal{N}_r)} = 0 \implies \mathcal{N}_r = \sqrt{\frac{\mathcal{A}}{\mathcal{S}_2}} \cdot \sqrt{\mathcal{N}_o}. \tag{5}$$

The above formula can be directly expanded for a number of candidates in C and the query point q,

$$\frac{\mathrm{d}(n_{\text{initial}})}{\mathrm{d}(\mathcal{N}_r)} = 0 \implies \mathcal{N}_r = \sqrt{(|C|+1) \cdot \frac{\mathcal{A}}{\mathcal{S}_2}} \cdot \sqrt{\mathcal{N}_o}. \tag{6}$$

N_r indicates the number of vertices on the initial terrain model and this resolution level is optimal in term of performance.

If all candidates can be ranked by the initial upper bounds (together with the lower bounds, discussed later), MR3 terminates; otherwise, next higher resolution level needs to be selected to compute more accurate upper bounds for the candidates not ranked yet. To do that, one straightforward approach is to select next resolution level in a fixed step for all remaining candidates. On the terrain model of this resolution level, the "wavefront" from the query point expands within the search region until all targeting candidates are reached. Since the upper bound computations for all candidates are conducted on single resolution terrain model, the network is traversed once. But, this approach processes all candidates equally even some of them are not in the final solution. To solve this problem, we proposed another approach motivated by the observation that k-NN query concerns the first k nearest neighbors ranking only. The candidates closer to the query point (i.e. their upper bounds are smaller based on previous computations) are more likely to be in the final solution and therefore they should be processed on a higher resolution terrain model in subsequent rounds of computation. In this approach, k candidates with minimum upper bound based on previous computation are processed first and called working candidates, collaboratively denoted as WCA. At the same time, other candidates are not processed until they are moved into WCA. That is, the more computing resource assigned in next round of iteration to the candidates which are more likely to be in the final solution, the less (or no) computing resource to other candidates. By doing this, the total performance in each round of iterations keeps same\similar. We call this scheme as "Equal Performance Scheme".

Since the initial round of computation is optimal in performance, we keep the performance in subsequent round not worse than it. To do that, the number of network vertices processed for next round is:

$$n_{\text{next}} = \sum_{i=1}^{k} S_{\text{next}}^{i} \frac{\mathcal{N}_{\text{next}}^{i}}{\mathcal{A}} = n_{\text{initial}}.$$

where $\mathcal{N}_{\text{next}}^{i}$ is the total number of vertices on the next terrain model for candidate $C[i]$ and S_i^{next} is the refined search region derived from the result of last iteration as described in Sect. 4.2.1.

For simplicity, the cost for transforming objects to be network vertices is ignored in the following analysis since the area of the triangular face keeps decreasing along with the increase of the terrain model resolution. For candidate $C[i]$, the next resolution can be estimated by:

$$\frac{N^i_{\text{next}}}{\mathcal{A}} \cdot S^i_{\text{next}} = \frac{n_{\text{initial}}}{k}.$$

Considering non-linear computation complexity of Dijkstra's algorithm, k can be ignored. Replacing n with Eq. (3), we have

$$N^i_{\text{next}} = \frac{\mathcal{S}_2}{S^i_{\text{next}}} \cdot \mathcal{N}_r. \tag{7}$$

Since the search region is refined in each round of iterations, the overall network access doesn't change while the resolution level increases. For $C[i]$, more and more accurate upper bound will be computed until $C[i]$ is identified in the final solution. If $C[i]$ cannot be identified using the highest accurate upper bound (i.e., on *pathnet*), $C[i]$ is replaced by the candidate with (k+1)th smallest upper bound in WCA.

For lb, the MSDN resolution level is usually discrete and evenly fixed at several levels for storage efficiency and simplicity. Although, for each iteration of lb computation, the search region is determined by the result of the current ub as described in Sect. 4.2.2, the application of dummy lower bound makes the lb computation cost trivial. Therefore, by starting from very low resolution, we compute dummy lb on all available MSDN resolutions until the minimal proper resolution is found.

5 Analysis

A cost model is constructed to estimate the candidate size in MR3. For the purpose of our algorithm analysis, we assume that objects are uniformly distributed on surfaces, and we also assume that, given a large region, there are numerous small mountains that are uniformly distributed and similar, i.e., each mountain has similar surface area and ground area (area on (x, y)-plane). Note that for the objects that are uniformly distributed on an individual mountain, it is obvious that their projections on (x, y)-plane may not be uniformly distributed. But from the view of the entire region (containing numerous mountains), the global distribution of all objects' projections on (x, y)-plane can be approximated to be uniform. These assumptions are reasonable and makes algorithm analysis possible. In MR3, all candidates are found in two steps. In the first step, given an object data set $\mathcal{D}$ and a query point q, let $C1[1..k]$ be the set of first k nearest neighbors of q based on d_E^{xy} (Euclidean distance between their projections on the (x, y)-plane) and $C1[i]$ is the ith nearest neighbor. Clearly, $C1$ is exclusively determined by the object coordinates (i.e. their locations on the (x, y)-plane). In $C1$, the maximum d_E^{xy} can be estimated by:

$$d_E^{xy}(C1[k], q) = \sqrt{k/(\pi \cdot \lambda)} \tag{8}$$

where λ is the number of objects whose projections in a unit area on (x, y)-plane as defined in Table 1. In the second step, the complete candidate set, denoted as $C2$, can be found based on $C1$. In $C1$, the maximum surface distance to the query point, $\max(d_S(C1, q))$, is used as the candidate filtering threshold. If d_E^{xy} of an object p $(p \in D - C1)$ to the query point is not longer than this threshold, p is a candidate since p may be closer to q than some objects in $C1$ when measured by surface distances (see Fig. 13b) for an example). Because the candidate filtering threshold is a surface distance, the cardinality of $C2$ depends not only on the coordinates of objects but also on the topography of the underlying surface around the query point. We use the following formula to estimate the complete candidate set:

$$|C2| = \pi \cdot \max{}^2(d_S(C1, q)) \cdot \lambda. \tag{9}$$

where $\max(d_S(C1, q))$ can be estimated by:

$$\max(d_S(C1, q)) = d_E^{xy}(C1[k], q) \cdot \delta. \tag{10}$$

thus, (9) can be transformed as:

$$\begin{aligned} |C2| &= \pi \cdot d_E^{xy}(C1[k], q)^2 \cdot \delta^2 \cdot \lambda \\ &= k \cdot \delta^2 \end{aligned} \tag{11}$$

where δ is the ratio between the maximum surface distance and the maximum Euclidean distance in $C1$. Equation (11) depicts that the complete candidate set increases when k and δ increase. If the surface is quite plain, the surface distance $d_S \approx d_E^{xy}$ such that $\delta \approx 1$ and $C1 \equiv C2$. But, if there are many mountains, the surface distance gets longer if the surface shortest path is affected by these obstacles. In this case, once the maximum surface distance in $C1$ gets longer, it is straightforward that δ increases and more candidates are contained in $C2$. On the other hand, if the maximum surface distance in $C1$ is not affected by these obstacles, $C2$ doesn't change.

6 Performance evaluation

In this section, we measure MR3 algorithm from several aspects: the response time, CPU time and the number of I/O, with varying values of k, object density λ and resolution scheme s (i.e., the resolution difference between two consecutive iterations).

6.1 Experiment Setup

The experiments are conducted on a PC (ADM Athlon XP 2400+ CPU, 1.3 GB memory). Two real world large scale terrain models, BH and EP, are created from USGS DEM files (data.geocomm.com) for two regions: Bearhead Mountain (WA) and Eagle Peak area (WY), USA. Both datasets cover an area around 10.7 km × 14 km and contain about 1.5 and 1.4 million elevation sample points respectively. The Bearhead area has more mountains than Eagle Peak. The object points are approximately uniformly distributed on the surface with varying object densities by setting $1 \leq \lambda \leq 10$ per km^2. DDM is built as previously described and the *pathnet* is created by inserting one Steiner point into each edge of the original terrain model. The MTM tree structure and the adjacent list of DDM are stored on the disk in the way as described in Sect. 3.4. Similarly, the MSDN is created and stored on the disk as well.

6.2 The benchmark algorithm

To calculate the surface distance, one can either use the exact algorithm or an approximate algorithm, as discussed in Sect. 2. The Chen and Han's algorithm is the only feasible exact surface distance algorithm. This algorithm can be used on the original terrain model to directly compute the surface distance. We test this approach (denoted as CH) using the implementation by Kaneva and O'Rourke [12]. An alternative approach is to use the Kanai and Suzuki algorithm for *ub* estimation where it starts from the original terrain model and continues to the *pathnet* level, and the 100% resolution SDN is used for *lb* estimation. We call this approach as the Enhanced Approximation Surface Distance Algorithm (EA). We allow 3% error in surface distance calculation (i.e., shortest surface distance range computation terminates once it reaches 97% accuracy). Figure 15 shows the performance of EA and CH. Clearly, CH is not scalable with the number of surface points. When a surface contains 10,000 vertices (that covers about 1km^2 in a 10 m elevation sample interval), this approach is practically not useable. Thus, EA is used as the benchmark algorithm in our experiments. For fair comparison, the methods used for finding the first global optimal shortest path and the search region refinement in the benchmark algorithm are the same as those used by MR3. Moreover, to highlight the effect of multi-resolution on *sk*-NN query processing, the benchmark algorithm also applies the same filtering techniques as MR3.

6.3 Distance range accuracy

A good indication of the effectiveness of multi-resolution surface distance model is how accurate the estimated *lb* and *ub* converges with the increase of LOD. Define accuracy $\varepsilon = \frac{lb}{ub}$, $0 \leq \varepsilon \leq 1$. A larger ε indicates a higher level of accuracy of *lb* and *ub* estimation. Figure 16 reports the accuracy with a range of MSDN and DMTM resolutions, where MSDN is represented as the percentage of the highest SDN resolution while the DMTM resolution is represented by the percentage of points comparing to that in the original resolution. Note that DMTM resolution 200% implies that a *pathnet* with one Steiner point per edge is used. At this level, $d_N = d_S$ by definition. The Euclidean distance is included as a way of estimating *lb* to show the effectiveness of *ub* estimation (i.e., with static *lb*). We observe that the best accuracy achieved is about 78% if the Euclidean distance is used as *lb* estimation. This is insufficient in most cases to differentiate the ranges of candidate points. On the other side, estimation accuracy can be improved rapidly and steadily when a higher LOD DMTM is used, for all SDNs. In the case of SDN resolution is 100%, MR3 eventually achieves 97% accuracy. With only 50% of DMTM used, the estimation accuracy already reaches 87%. Using the MR3 approach, a query like "what is the surface distance between *a* and *b* within accuracy 95%" can be directly processed. This level of accuracy is sufficient for most applications we consider. It is possible to achieve an even higher level of accuracy (by simply inserting more Steiner points into the highest LOD terrain model to generate DMTM at higher resolution). But the cost is too high.

In Sect. 6.4 and 6.5, we test equal performance scheme and three sets of fixed step length resolution scheme as given below:

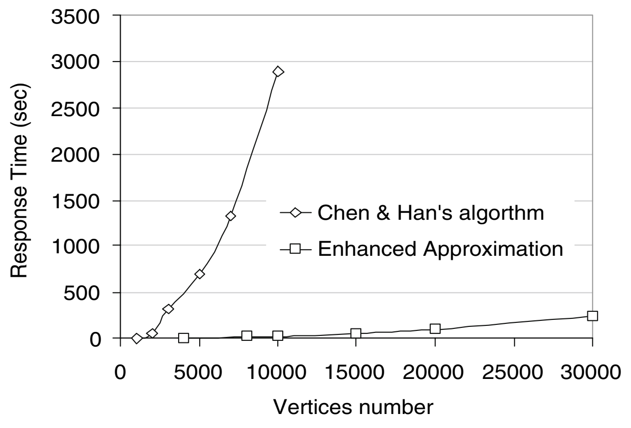


Fig. 15 Algorithm CH versus Algorithm EA

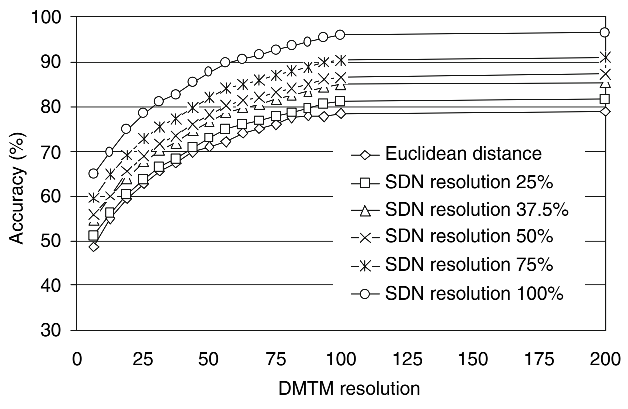


Fig. 16 Distance range accuracy

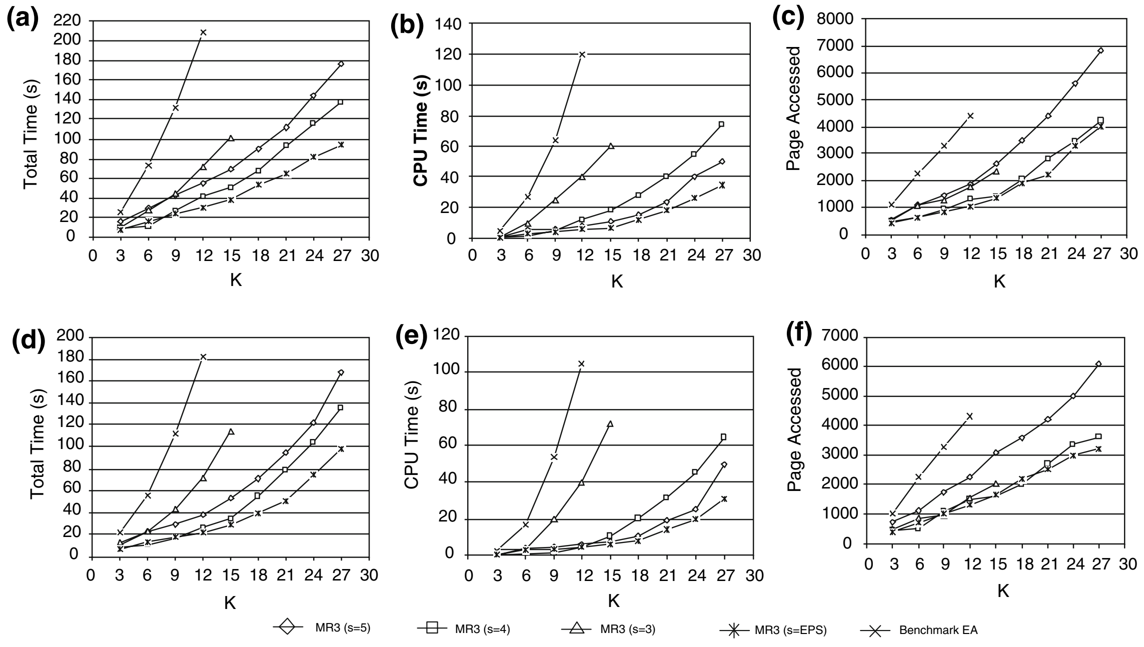


Fig. 17 Effect of k, using dataset BH (**a**–**c**) and EP (**d**–**f**) where $\lambda = 4$

1. $s = 5$: DMTM: 0.5%, 25%, 50%, 75%, 100%, 200%; MSDN: 25%, 37.5%, 50%, 75%, 100%;
2. $s = 4$: DMTM: 0.5%, 50%, 100%, 200%; MSDN: 25%, 50%, 100%;
3. $s = 3$: DMTM: 0.5%, 100%, 200%; MSDN: 25%, 100%; and
4. $s =$ EPS: equal performance scheme

The fixed resolution levels are used in $s = 5$, $s = 4$, $s = 3$. For example, $s = 3$ means the experiment for the upper bound computation begins from 0.5% of the original resolution and next higher resolution is 100% of that. After that, the *pathnet* is used. At the same time, the resolution of MSDN starts from 25%, then jumps to 100%. $s =$ EPS means the equal performance resolution scheme is used as described in Sect. 4.3. For each object, the initial resolution is estimated by Eq. (6) and the following resolutions are estimated by Eq. (7) for working candidates.

6.4 Effect of k

This set of experiments is to test the performance of MR3 with varying k value (from 3 to 30) with $\lambda = 4$.

Figure 17a–c illustrate the experiment results for dataset Bearhead (BH) which has more mountains than that of dataset Eagle Park (EP) whose experiment results are presented in Fig. 17d–f. As we discussed in Sect. 5, the effect of obstacles in the vicinity of the query point on the terrain surface may impact the parameter δ. Generally, more mountains lead to a longer surface distance. So, the BH has a greater δ than EP in average. By Eq. (11), the cardinality of the candidate set is decided by the value of k and δ. When k is fixed, the greater δ means more objects need to be processed. Figure 17a–c shows a better performance than that of Fig. 17d–f under the same circumstances.

The comparison of performance between MR3 and benchmark EA is also depicted in Fig. 17. Despite the impact of the varying resolution scheme, the overall test results of MR3 outperform the benchmark remarkably in total time and CPU time. As depicted in Fig. 17a, the total time of EA increases very rapidly so that it is not practical when $k \geq 12$. On the other side, MR3 shows a much slow increase rate when k increases from 1 to 30. When $s =$ EPS, MR3 outperforms EA by almost one order of magnitude. When $s = 3$, MR3 has a performance increase pattern more similar to EA comparing to the others. This is because the case of $s = 3$ is less multi-resolution supported (this simulates the traditional filter-and-refine approach that jumps to the full resolution data after one filtering step).

To compare the different resolution schemes, we need to understand how the performance is affected by the selection of the resolutions. In general, a large jump to higher resolution implies less iteration needed for sk-NN query processing. However, this also means less opportunity to use tighter distance bound estimation (1) to reduce the size of search region and (2) to terminate the search earlier. Since the search region of next iteration is decided by previous computation, if higher resolution is selected, this region will contain more surface data need to be accessed and processed. Due to the $O(n^2)$ time complexity of Dijkstra's algorithm,

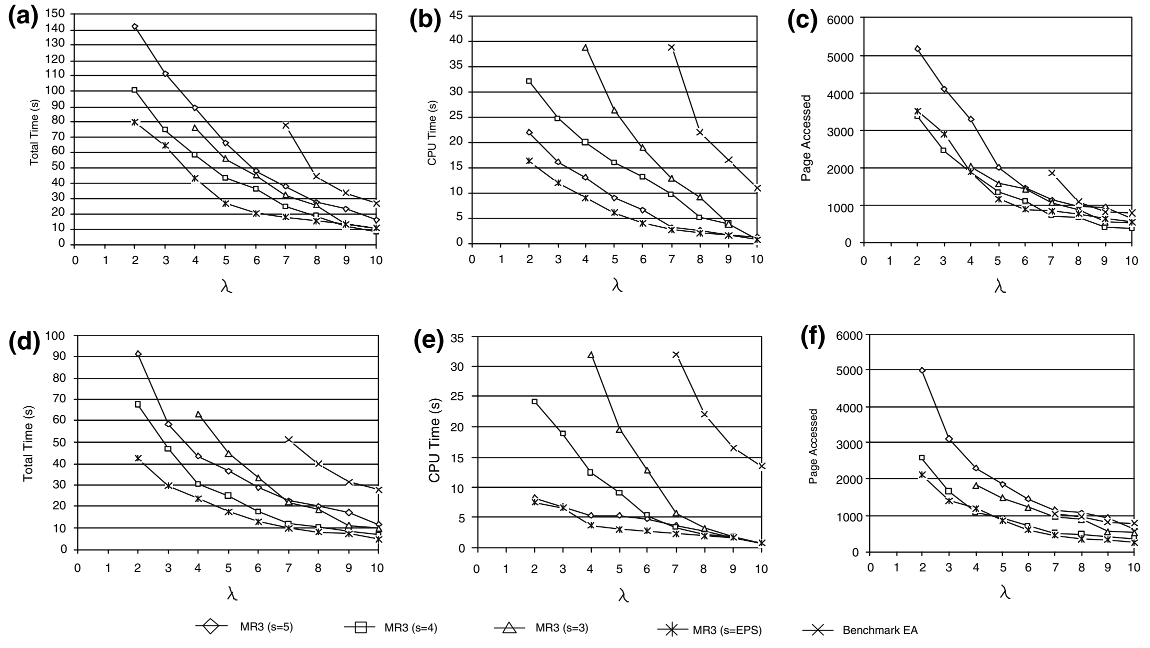


Fig. 18 Effect of object density using dataset BH (**a**–**c**) and EP (**d**–**f**) where $k = 10$

if the number of network vertices is, for example, doubled, the cost to compute the network distance will increase non-linearly. In the cases of $s = 4$ and $s = 5$, since the search regions keep refined before a high resolution terrain model is used, the number of network vertices processed in this higher resolution level is smaller than that in $s = 3$. That is a major reason why the $s = 3$ is the worst case in terms of both CPU and total time. Another reason is that $s = 3$ loses chance to identify the ranking of some objects in lower resolution levels. In the situation of small resolution jump step, the total network vertices processed may be reduced as $s = 4$ (shown in Fig. 17c–f). However, too small jump step (implies too many iterations) will increase the total number of network vertices accessed and processed, e.g., $s = 5$. Note that there is one more iteration from $s = 4$ to $s = 3$ and two more iteration from $s = 5$ to $s = 4$, but the performance improvement in former situation is much more significant than the latter case. This is due to the increase of the total network vertices accessed and processed in $s = 5$ when more iterations are applied (see Fig. 17c–f).

We observe that $s =$ EPS outperforms all the fixed resolution schemes. This superiority is a result of the application of equal performance scheme where the computing resource is only assigned to the candidates more likely in the final solution.

6.5 Effect of λ

Now we test the effect of object density, by fixing $k = 10$. In general, the cost reduces as the object density increases. That is, on the same terrain model for a given k, high object density λ leads to a small search region so that less surface data will be retrieved and processed. This is proved by the experiment results in Fig. 18. We observe that the benchmark EA illustrates a quick increase when λ decreases. This is because EA starts *sk*-NN processing from the original terrain model, by which the search and I/O region are not fully optimized by any multi-resolution technique. In the contrary, benefited from using multi-resolution surface distance model, the overall performance of MR3 for all resolution schemes is significantly better than that of EA; and the best case is when $s =$ EPS. More detailed explanation about the effects of different resolution schemes are the same as that in Sect. 6.4.

6.6 Candidate size

Finally, we examine the candidate size in various settings for k and δ (d_S/d_E^{xy}). Figure 19a shows the increase rate of the candidate size against δ is non-linear while Fig. 19b illustrates the candidate size is proportional to the value of k. Clearly, these results comply with the analysis result in Eq. (11).

7 Conclusion

This paper is a first in-depth study on efficient *sk*-NN query processing. A multi-resolution surface distance model is proposed to guarantee the monotonic distance change property when distances are computed at different resolution levels.

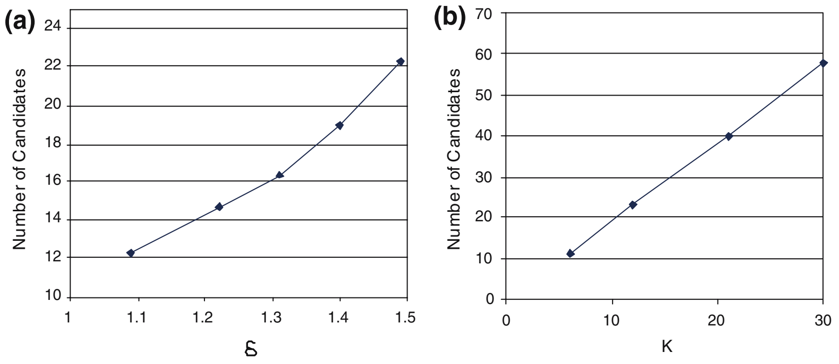


Fig. 19 **a** Candidate size versus δ ($k = 10$), **b** Candidate size versus k ($\delta = 1.4$)

This important property makes it possible to minimize the amount of surface data access as well as the cost for surface distance computation. Our sk-NN query processing algorithm, MR3, can therefore access and process the terrain data in a just-enough manner. Our experiments using large scale real terrain data have shown that our approach outperforms the benchmark algorithm in all cases by up to one order of magnitude.

Our previous work on developing MTMs for efficient visualization [26] has been extended to support surface k-NN query processing. The new multi-resolution surface distance model is a framework capable of supporting other types of spatial queries that require distance comparison as part of its query processing strategy, such as range queries and closest pair queries. The idea of progressive accuracy increase of surface distance estimation is also applicable to other types of surface-based queries with a specified target ROI and LOD. Next on our research agenda, we will investigate the modelling and query processing techniques towards efficient sk-NN query processing with obstacle constraints, which can be found in many real-life sk-NN applications, such as energy consumption and vehicle stability considerations for rovers, and general traversability constraints.

Acknowledgements The work reported in this paper has been partially supported by grant DP0345710 from the Australian Research Council.

References

1. Chen, J., Han, Y.: Shortest paths on a polyhedron. In: 6th ACM Symp. Comput. Geometry, pp. 360–369 (1990)
2. Deng, K., Zhou, X.: Expansion-based algorithms for finding single pair shortest path on surface. In: Proc. of W2GIS, pp. 254–271 (2004)
3. Deng, K., Zhou, X., Shen, H.T., Xu, K., Lin, X.: Surface k-NN query processing. In: ICDE (2006)
4. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)
5. Garland, M.: Multiresolution modeling: survey and future opportunities. In: Eurographics, pp. 111–131 (1999)
6. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: 24th Int'l Conf. on Comput. Graphics and Interactive Tech. pp. 209–216 (1997)
7. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. TODS **24**(2), 265–318 (1999)
8. Hoppe, H.: Progressive meshes. In: SIGGRAPH (1996)
9. Jagadish, H., Ooi, B., Tan, K.L., Yu,C., Zhang, R.: iDistance: an adaptive B^+-tree based indexing method for nearest neighbour search. TODS (2005)
10. Jiang, B.: I/O efficiency of shortest path algorithms: an analysis. ICDE (1992)
11. Kanai, T., Suzuki, H.: Approximate shortest path on polyhedral surface based on selective refinement of the discrete graph and its applications. Geom. Model. Process. 241–250 (2000)
12. Kaneva, B., O'Rourke, J.: An implementation of Chen & Han's shortest paths algorithm. In: Proc. of 12th Canadian Conf. on Comput. Geom. pp. 139–146 (2000)
13. Kapoor, S.: Efficient computation of geodesic shortest paths. In: 31st Annual ACM Symp. on Theory of Computation, pp. 770–779 (1999)
14. Kolahdouzan, M.R., Shahabi, C.: Voronoi-based k nearest neighbor search for spatial network databases. VLDB (2004)
15. Li, Z., Openshaw, S.: Algorithms for automated line generalization based on a natural principle of objective generalization. J. GIS **6**(5), 373–389 (1992)
16. Mitchell, J.S.B.: Geometric shortest paths and network optimization. In: Handbook of Computational Geometry, Sack, J.-R., Urrutia, J. (eds) pp. 633–701 (2000)
17. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. VLDB (2003)
18. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. SIGMOD (1995)
19. Seidl, T., Kriegel, H.P.: Optimal multi-step k-nearest neighbor search. SIGMOD (1998)
20. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. In: ACM GIS, pp. 94–100 (2002)
21. Shekhar, S., Kohli, A., Coyle, M.: Path computation algorithms for advanced traveler information system (atis). ICDE (1993)
22. Shekhar, S., Liu, D.: A connectivity-cluster access method for networks and network computations. TKDE **19**(1), 102–119 (1997)
23. Tompkins, P., Stentz, T., Whittaker, W.: Mission planning for the sun-synchronous navigation field experiment. In: IEEE Int'l Conf. on Robotics and Automation, pp. 3493–3500 (2002)
24. Varadarajan, K.R., Agarwal, P.: Approximating shortest paths on an nonconvex polyhedron. In: Proc. 38th Annu. IEEE Symp. Found. Comput. Sci. pp. 182–191 (1997)
25. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. VLDB (1998)
26. Xu, K., Zhou, X., Lin, X.: Direct mesh: a multiresolution approach to terrain visualisation. ICDE (2004)
27. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. TKDE **17**(6), 820–833 (2005)