July 2006

Fast Algorithms Using Minimal Data Structures for Common Topological Relationships in Large, Irregularly-spaced Topographic Data Sets

Thomas H. Meyer
University of Connecticut Department of Natural Resources Management and Engineering

Fast algorithms using minimal data structures for common topological relationships in large, irregularly-spaced topographic data sets

Thomas H. Meyer [a,*],

[a] *University of Connecticut*

Department of Natural Resources Management and Engineering

Storrs, CT 06269-4087, USA

Abstract

Digital terrain models (DTM) typically contain large numbers of postings, from hundreds of thousands to billions. Many algorithms that run on DTMs require topological knowledge of the postings, such as finding nearest neighbors, finding the posting closest to a chosen location, etc. If the postings are arranged irregularly, topological information is costly to compute and to store. This paper offers a practical approach to organizing and searching irregularly-space data sets by presenting a collection of efficient algorithms ($O(\mathrm{N}), O(\lg \mathrm{N})$) that compute important topological relationships with only a simple supporting data structure. These relationships include finding the postings within a window, locating the posting nearest a point of interest, finding the neighborhood of postings nearest a point of interest, and ordering the neighborhood counter-clockwise. These algorithms depend only on two sorted arrays of two-element tuples, holding a planimetric coordinate and an integer identification number indicating which posting the coordinate belongs to. There is one array for each planimetric coordinate (eastings and northings). These two arrays cost minimal overhead to create and store but permit the data to remain arranged irregularly.

Key words: Digital terrain model, irregularly-spaced data, topological relationships, Triangulated Irregular Network, TIN

1 Introduction

Topographic data sets are sets of triplets containing two planimetric coordinates and one vertical coordinate. These coordinates are either measured by automatic methods such as scanning laser altimeters (LIDAR) (Flood and Gutelius, 1997; Baltsavias, 1999), interferometric synthetic aperture radar (IFSAR) (Hodgson et al., 2003; Gamba and Houshmand, 2000; Mercer and Schnick, 1999), or by manual compilation with methods like photogrammetry and ground surveying. The terrain samples are called **postings**. Automatic terrain sampling methods produce irregularly-spaced samples either by design or simply due to uncontrollable environmental factors such as wind turbulence jostling the aircraft carrying an instrument. Samples collected by manual methods are frequently arranged irregularly by choice in order to capture breaklines and other important features that define the shape of the topography; irregularly-spaced postings capture the shape of the terrain and the features thereon better than gridded postings (Makarovic, 1977; Gould, 1981; Douglas, 1986). Additionally, some applications require irregularly spaced data. For example the U.S. National Geodetic Survey maintains a database of high-accuracy survey control markers and provides web-based applications

* Corresponding author. Tel: +1-860-486-0145. Fax: +1-860-486-5408.
Email address: `thomas.meyer@uconn.edu` (Thomas H. Meyer).

that allow a user to query the database to find all markers within a certain distance of a point of interest. The published coordinates of these markers must not be changed by their representation in the database; they must remain irregularly spaced. Also, gridding the data can impede feature detection (Cooper and Cowan, 2004). Digital terrain models employing irregularly spaced postings are common and useful; the Triangulated Irregular Network (TIN) is probably the most common example of the type.

Many terrain analysis algorithms depend on topological relationships between the postings. In particular, many algorithms require **neighborhoods** of postings that are close to one another in some sense. For example, the computation of gradients (Meyer et al., 2001), curvature (Shary, 1995; Ozkaya, 2002), semivariograms (Isaaks and Srivastava, 1989), kriging (Hessami et al., 2001), roughness metrics (Philip and Watson, 1986), cluster analysis (Gebhardt, 2001), feature recognition (Cooper and Cowan, 2004), and fractal dimensions (DeCola, 1989; Cheng, 1999) are defined over neighborhoods. For gridded data, two typical neighborhoods are the four cardinal postings around the point of interest or the cardinal postings plus the diagonals. For irregularly spaced data, the situation is less clear. One popular way to determine sets of nearest neighbors for irregularly spaced data is to construct the Delaunay tessellation of the postings. Then, for some posting p, take the nearest neighbors of p to be those postings that share an edge in the tessellation with p. This solution is elegant and satisfies the goal of "letting the data speak for themselves" (Gould, 1981), but a Delaunay tessellation requires considerable time to compute and space to store. These problems can be intractable given the size of many topographic data sets. For example, as of the time this article was written, at least one commercial LIDAR sensor can collect samples at 70,000 Hz with sub-meter posting spacing (Optech, 2003). At this rate, a one-hour flight of this sensor would collect more than $2.5{\times}10^8$ samples.

The time needed to compute the inherent topology of data sets as large as these would be prohibitive to most users. Therefore, large topographic data sets are usually gridded and the resulting loss of accuracy is simply accepted. This paper offers an alternative, a way to have the accuracy of irregularly spaced data without unacceptable computational and storage burdens of complicated data structures such as Delaunay tessellations (Mortenson, 1985, p. 317), quadtrees (Samet, 1990; de Berg et al., 1998), k-d-B-Trees (Bentley, 1975; Robinson, 1981), hB-Trees (Lomet and Salzberg, 1989, 1990) or R-Trees (Guttman, 1984); see Nievergelt and Widmayer (1991) for a survey. This paper presents several simple and efficient algorithms that compute the basic topological relationships needed for algorithms requiring neighborhoods for inputs. These algorithms depend only on two simple data structures, namely, two sorted arrays.

2 Supporting Data Structure

The following discussion depends on sets and the elements thereof. The i^{th} element of a set P is denoted P_i. Conversely, we denote that element itself with p^i. Thus, $P_i = p^i$.

An individual postings is typically a set of values including three spatial coordinates plus other ancillary information such as an intensity value, a time stamp, a return number, etc. Define a posting to be a set $p^i = \{e^i, n^i, u^i, \alpha^i, \beta^i, \ldots\}$, where $e^i, n^i, u^i \in \mathbb{R}$ are the posting's easting, northing, and height (up) coordinates, respectively, $\mathbb{R}$ denotes the set of reals, and α^i, etc. are additional attribution fields holding ancillary information of no particular type. Let p^i_e, p^i_n, p^i_u denote the easting, northing, and up coordinate of posting p^i and $P = \{p^1, \ldots, p^{\mathrm{N}}\}$ denote the given posting data set. Thus, $P_{i,e}$ is the easting of p^i. Define the index set over P to be $I = \{1, \ldots, \mathrm{N}\}$.

Our strategy is to decompose P into three arrays. One of the arrays, $\mathcal{N}$, is a sorted array of northings together with an index indicating which posting that northing came from. $\mathcal{E}$ is a sorted array of eastings together with an index into $\mathcal{N}$ indicating which northing that easting was paired with. The last array, $\mathcal{P}$ contains the attribution fields of P plus an index into $\mathcal{E}$ thus forming an index loop: knowing an easting leads to the northing associated with that easting; knowing a northing leads to the attribution information and a pointer to the associated easting; and knowing a posting leads to its easting. Therefore, given any tuple in any of $\mathcal{E}, \mathcal{N}, \mathcal{P}$ allows the entire original posting to be reconstructed. For notation, let $\mathcal{N}_i = \{n^i, \eta^i\}$, meaning $\mathcal{N}_i$ is the i^{th} tuple of the sorted northing array, n^i is the northing coordinate and η^i is the index of the posting having this northing. Similarly, let $\mathcal{E}_i = \{e^i, \epsilon^i\}$, meaning $\mathcal{E}_i$ is the i^{th} tuple of the sorted easting array, e^i is the easting coordinate and ϵ^i is the index into $\mathcal{N}$ of the northing associated with this easting. Finally, let $\mathcal{P}_i = \{\pi^i, \alpha^i, \beta^i, \ldots\}$, meaning $\mathcal{P}_i$ holds the attribution information of the i^{th} posting and π^i is an index into $\mathcal{E}$ indicating that $\mathcal{E}_{\pi^i,e}$ is the easting of posting i. For example, suppose the original posting set consisted only of eastings, northings, and elevations (no additional attribution fields):
$P = \{\{170, 430, 10\}, \{100, 400, 0\}, \{130, 440, 20\}, \{120, 410, 50\}\}$. This posting set is split into three pieces,
$\mathcal{N}$=\{ \{400,2\},\{410,4\},\{430,1\},\{440,3\} \},

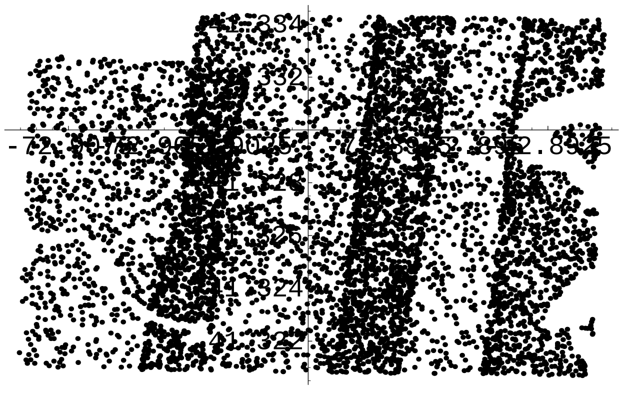

Fig. 1. 7000 random postings from the 3 627 915 posting data set used to test these algorithms. The sinuous dropout on the right is from a waterway.

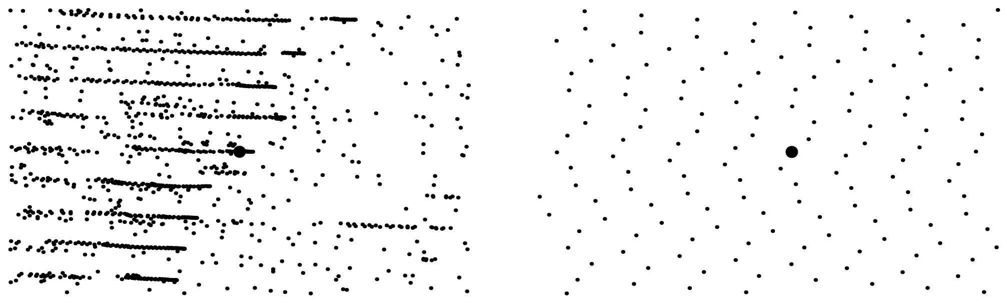

Fig. 2. Two detailed views of posting spacings. The left image shows the ends of several scan lines that overlay other scans roughly at their nadir point (the heavy dot in the center). The highly heterogeneous pattern is due to overlapping nominally-orthogonal scan lines, topographic variability, and land cover preventing some laser beams from reaching the ground. The right image has no overlap and is typical of spacings near the nadir.

$\mathcal{E} = \{\ \{100,1\},\{120,2\},\{130,4\},\{170,3\}\ \}$, and

$\mathcal{P}=\{\ \{4,10\},\{1,0\},\{3,20\},\{2,50\}\ \}$.

2.1 Expectations of Redundancy

In 2005, the University of Connecticut obtained a multi-return LIDAR data set covering part of the Connecticut coast on Long Island Sound. A subset containing 3 627 915 postings was extracted to test these algorithms. 7000 random postings from this data set are shown in Fig. 1, which gives a general impression of the arrangement of the 3.6 million postings. Fig. 2 shows two detailed subsets, to illustrate the posting spacing variety.

The algorithms in this paper were implemented in *Mathematica* v5.1 running on a Dell Optiplex GX260, 2.40 GHz CPU with 512MB RAM and the data structures were constructed using external storage. Many of the algorithms that follow have a linear time complexity component so it is useful to note that scanning the data required 27.9 minutes, for the average single-posting retrieval speed is about 460 μs. This is the concrete upper bound for a linear complexity algorithm.

Examining this data set revealed that many postings have identical eastings and/or northings. This appears to have happened because this LIDAR uses a “whisk broom” beam steering mechanisms that sweeps the laser perpendicularly across the flight line and most whisk broom sensors slow down at the end of the sweep causing postings to “pile up” at the ends (Fig. 2). Of the 3 627 915 postings, only 183 306 have distinct eastings and 127 384 have distinct northings, with as many as 97 postings sharing a common northing coordinate. This redundancy occurred in spite of the coordinates, given in geographic decimal degrees, being reported to seven significant decimal digits. Furthermore, 8571 coordinate pairs (0.26%) were not unique. In fact, there are six cases in which nine postings all have the same easting-northing pair. Therefore, we assume that $\mathcal{E}$ and $\mathcal{N}$ contain tuples with replicated planimetric coordinates but different indices. For example, suppose the original posting set was (note duplicated coordinates)

$P = \{\{130, 430, 10\}, \{160, 400, 0\}, \{130, 400, 20\}, \{100, 490, 50\}\}$. This posting set is split into

$\mathcal{N}$={ {400,2},{400,3},{430,1},{490,4} },

$\mathcal{E}$ = { {100,4},{130,3},{130,2},{160,1} }, and

$\mathcal{P}$={ {2,10},{4,0},{3,20},{1,50} }.

3 Fundamental Topological Relationships

In what follows we assume the availability of a binary search algorithm (Cormen et al., 1997) that will be used to search over the sorted coordinates in $\mathcal{E}$ and $\mathcal{N}$. We name this algorithm, "BinarySearch." We assume it takes two operands. The first is the name of the sorted array over which to search, either $\mathcal{E}$ or $\mathcal{N}$. The second operand is a value t to search for among the planimetric coordinates. BinarySearch returns the index associated with the two-tuple of either $\mathcal{E}$ or $\mathcal{N}$ whose planimetric coordinate is closest to t. We note that this algorithm must be able to return a list of indices, not just a single value. This is true because, as was noted above, it's possible that the closest coordinate value could belong to more than one point. Furthermore, t could fall exactly between a set of points, such as finding a perpendicular bisector or a point in the center of a grid cell.

We now present algorithms for computing various topological relationships given $\mathcal{E}$, $\mathcal{N}$ and BinarySearch.

3.1 Bounding Rectangle and Geometric Center

The bounding rectangle is simply $(\mathrm{First}[\mathcal{E}], \mathrm{First}[\mathcal{N}]), (\mathrm{Last}[\mathcal{E}], \mathrm{Last}[\mathcal{N}])$, where First and Last are functions that extract the first and last coordinate from their argument, respectively. The geometric center is the average of the bounding rectangle coordinates.

3.2 Square Window

The following is an algorithm to return the set of indices $\mathcal{I}_{\Box} \subseteq I$ into P of postings $P_{\Box} \subseteq P$ that are inside a square window having sides of length $2r$ and centered at $p = \{e, n\}$. p may or may not be in P. The algorithm depends on the following claim.

Let ϵ^+ be the largest index into $\mathcal{E}$ such that $\mathcal{E}_{\epsilon^+}$ has the largest easting less than or equal to $e + r$. Symmetrically, let ϵ^- be the smallest index into $\mathcal{E}$ such that $\mathcal{E}_{\epsilon^-}$ has the smallest easting greater than or equal to $e - r$. Define η^+ and η^- similarly on $\mathcal{N}$. Let $\mathcal{E}_{[\epsilon^-,\epsilon^+]}$ denote the set of elements of $\mathcal{E}$ in the range $[\epsilon^-, \epsilon^+]$, inclusive, and ϵ to be the set of indices of $\mathcal{E}_{[\epsilon^-,\epsilon^+]}$ into $\mathcal{N}$. Then the required set of indices is equal to

$$\mathcal{I}_{\Box} = \epsilon \bigcap [\eta^-, \eta^+].$$

A set of irregularly spaced postings, indicated with open circles, is depicted in Figure 3 with the point of interest p shown as a solid circle. $\mathcal{E}_{[\epsilon^-,\epsilon^+]}$ is the index set of those circles in the darkened vertical region. $[\eta^-, \eta^+]$ is the index set of those circles in the darkened horizontal region. The white intersection of the two is the set of postings common to both sets, those points in the required

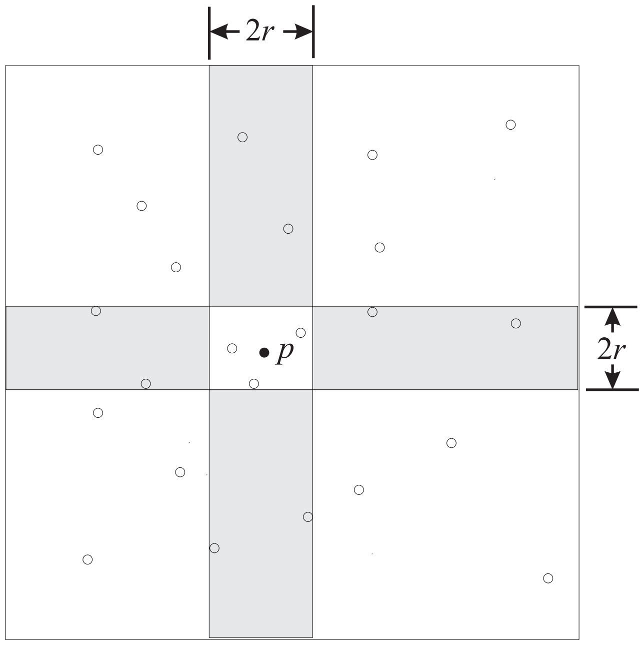

Fig. 3. Postings within a square window with size $2r$ centered at p, shown as a solid black circle. The darkened vertical region contains those postings in $\mathcal{E}_{[\epsilon^-,\epsilon^+]}$ and the darkened horizontal region contains those postings in $[\eta^-, \eta^+]$. The intersection of the two are the three postings satisfying both conditions and are, therefore, those postings in the required window.

window.

The algorithm to compute $\mathcal{I}_\Box$ is as follows. Find the two indices into $\mathcal{E}$, ϵ^- and ϵ^+, for the lower/upper easting bounds. All postings with easting index between these bounds have eastings within the required range. Find the two indices into $\mathcal{N}$, η^- and η^+, for the lower/upper northing bounds. Recall that $\mathcal{E}_{i,\epsilon}$ is an index into $\mathcal{N}$ and that, because $\mathcal{N}$ is sorted, any index into $\mathcal{N}$ between η^-, η^+ must be a northing that falls within the required northing range. Therefore, perform the intersection by scanning the northing indices in $\mathcal{E}$ for those between η^-, η^+. The indices of the postings $\mathcal{I}_\Box$ are then found from $\mathcal{N}$.

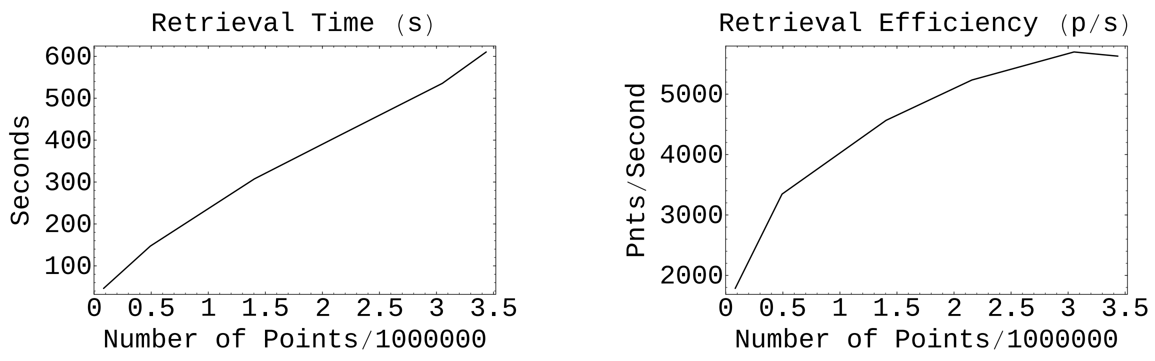

Fig. 4. (a) Time to find all postings falling within square windows ranging in size from 80,000 to 3.4 million postings. (b) Retrieval efficiency (postings per second) for windows of various sizes.

For a dataset with N postings, the computational complexity of $\mathcal{I}_\Box$ is $O(N)$ because, if the window were to encompass the entire data set, all postings would have to be examined. However, the intersection can be performed by a linear scan of only $\mathcal{E}$ because knowing η^- and η^+ allows each candidate from $[\eta^-, \eta^+]$ to be considered without actually scanning $\mathcal{N}$. Fig. 4 presents two graphs showing the running time of our implementation. The linear complexity is evident in Fig. 4(a) but notice that the maximum running time was around 10 minutes, which is more than twice as fast as a linear scan of the postings. Fig. 4(b) shows the *efficiency* of the algorithm in terms of postings retrieved per second. Interestingly, the efficiency is better than constant; it increases with window size. This is simply explained by the fact that more postings satisfying the individual easting / northing range end up in the answer whereas, with a smaller window, many more postings are examined to be discarded.

This algorithm can be generalized to windows such as circles, rectangles, or polygons simply by finding the bounding square of the generalized window, executing the given algorithm on that square, and passing the results through

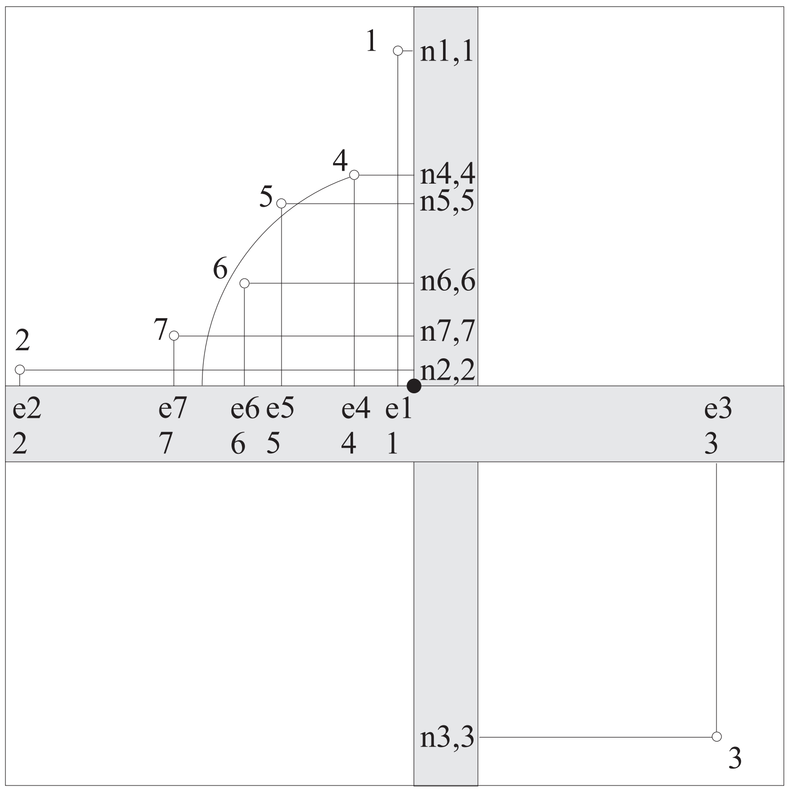

Fig. 5. A sample posting set to illustrate the nearest-posting algorithm. Postings are denoted with open circles. The point of interest p is the solid circle near the middle. $\mathcal{E}$ is shown as the horizontal shaded area. $\mathcal{N}$ is shown as the vertical shaded area. Note that posting 6 is closest to p but the eastings and northings closest to those of p belong to points on the data set's convex hull, at extreme distances from p.

a filter to remove those postings outside the generalized window.

3.3 Postings Nearest a Point of Interest

Suppose it is required to find a posting that is spatially closest to some point of interest $p = \{e, n\}$, and p is typically not in P. There is usually only one posting nearest p but, as stated above, that need not be the case. The following is an algorithm to return the set $\mathcal{I}_N \subseteq I$ of indices into P of postings $P_N \subseteq P$

nearest to a point of interest p. Logically, we define $\mathcal{I}_N$ to be the set of indices for those postings whose distance from the point of interest is less than that for all other points in P. Conceptually, one could sort the points by distance to p and simply take the shortest one, or all points sharing the shortest distance.

The algorithm works using the common sense notion that the posting closest to p must have its entries in $\mathcal{E}$ and $\mathcal{N}$ close to those found by searching for p. It is possible that the search will find the closest point directly but this need not be true. See Figure 5. The solid circle in the center represents p's location. The gray regions are $\mathcal{E}$ and $\mathcal{N}$. Posting 6 is closest to p but there are two or three entries in $\mathcal{N}$ and $\mathcal{E}$ respectively that are closer. In fact, it is possible for the location of one of the closest posting's coordinates in $\mathcal{N}$ and $\mathcal{E}$ to be arbitrarily far away from the coordinate found with the binary search. As suggested by the figure, there could be a cascade of other postings whose easting, say, were closer. However, if one coordinate is far away, the other cannot be. This is guaranteed by the triangle inequality. Therefore, searching $\mathcal{E}$ and $\mathcal{N}$ in all four directions simultaneously is guaranteed to find the closest posting quickly. The algorithm is organized as follows.

First: Find the index sets $\mathcal{E}^\circ \subseteq I$ and $\mathcal{N}^\circ \subseteq I$ of the postings whose easting and northing coordinates are closest to e and n by performing a binary search of $\mathcal{E}$ and $\mathcal{N}$. That is, $\mathcal{E}^\circ = \text{BinarySearch}(\mathcal{E}, e)_\epsilon$ and $\mathcal{N}_\circ = \text{BinarySearch}(\mathcal{N}, n)_\eta$. Let $\iota^\circ = \mathcal{E}^\circ \bigcap \mathcal{N}^\circ$.

Claim: $\iota^\circ \neq \emptyset \Rightarrow \iota^\circ$ is the set of the indices of the closest postings.

Proof: First, suppose $\iota^\circ \neq \emptyset$. Claim: the posting(s) in ι° are the closest. There is no posting whose easting is closer to e than those in $\mathcal{E}^\circ$. Likewise, there is no posting whose northing is closer to n than those in $\mathcal{N}^\circ$. Then $P_{\iota^\circ} = P_{\mathcal{E}^\circ} \bigcap P_{\mathcal{N}^\circ}$

is not empty by assumption and is exactly the set of postings closest to p because there are no postings whose easting is closer to e than those in $P_{\mathcal{E}^\circ}$ and there are no postings whose northing is closer to n than those in $P_{\mathcal{N}^\circ}$. Thus, P_{ι° contains all the closest postings and every posting in P_{ι° belongs there. ■

The algorithm works as follows:

If the intersection is not empty, the search is completed.

Now suppose that $\iota^\circ = \emptyset$. Let p^i and p^j be the postings whose easting and northing coordinates are closest to those of p, respectively. The lesser of the two distances $|p^i - p|$ and $|p^j - p|$ is an upper bound $\bar{d}$ on how far the closest point $p^* = \{e^*, n^*\}$ can be from p. This implies that $|e - e^*| \leq \bar{d}$ and also $|n - n^*| \leq \bar{d}$ (triangle inequality). We now show how to efficiently find p^* using $\mathcal{E}$ and $\mathcal{N}$. The efficiency comes from noting that there must be a (always proper) subset of $\mathcal{E}$ and $\mathcal{N}$ in which p^* must reside, and this subset is usually far smaller than P. Therefore we will search $\mathcal{E}$ and $\mathcal{N}$ to find p^* using $\bar{d}$ as an initial bound on the search. Recall that $\mathcal{E}^\circ$ is a set of indices of postings whose easting is closest to p's easting; similar for $\mathcal{N}^\circ$. Both $\mathcal{E}^\circ$ and $\mathcal{N}^\circ$ will usually actually have only one element but there could be more if more than one posting's easting/northing coordinate were identical and also closest to p. Define ϵ° to be any index from $\mathcal{E}^\circ$ and η° to be any index from $\mathcal{N}^\circ$. The postings associated with ϵ° and η° are not closest because, by assumption, the intersection of $\mathcal{E}^\circ$ and $\mathcal{N}^\circ$ was empty. Therefore, begin the search simultaneously at two locations in $\mathcal{E}$ and $\mathcal{N}$ each, namely $\epsilon^\circ \pm 1$ and $\eta^\circ \pm 1$. The postings associated with these four elements of $\mathcal{E}$ and $\mathcal{N}$ may be nearer or farther from p^* than the original points were from p^*. However, if any of the four are closer, denote the closest by $\hat{p}^*$ and that distance by $\hat{d}^*$ which becomes a new, better upper bound on the search. This reduces the range of $\mathcal{E}$ and $\mathcal{N}$ that must be searched because we need look no further away from ϵ° or η° than $\hat{d}^*$. The search continues looking at elements of $\mathcal{E}$ and $\mathcal{N}$ incrementally further from ϵ° and η° until the coordinates are farther from ϵ° or η° than $\hat{d}^*$, after which the search terminates. The critical observation is that a posting with a coordinate that is itself further from p than $\hat{d}^*$ cannot possibly be the closest posting because that coordinate by itself is already too far away. Furthermore, since $\mathcal{E}$ and $\mathcal{N}$ are sorted by coordinates, we know we can stop the search because all subsequent postings must be further away than $\hat{d}^*$ for the same reason. At this point we know $\hat{d}^* = \bar{d}$. Note that each iteration in any of the four directions can potentially reduce the block size for all the searches. Thus, at each step, more information can be gained to shrink the search space. The pseudo-code for this algorithm is as follows.

```
$\mathcal{I}_N(p)\{$
  N = number of postings;
  $\epsilon^\circ :=$ BinarySearch$[\mathcal{E},\ e]_\epsilon$; $\eta^\circ :=$ BinarySearch$[\mathcal{N},\ n]_\eta$;
  $\iota^\circ := \mathcal{E}_{\epsilon^\circ,\epsilon} \bigcap \mathcal{N}_{\eta^\circ,\eta}$;
  if $\iota^\circ \neq \emptyset$
    $\mathcal{I}_N := \iota^\circ$;
  else
    $\bar{d} := \infty$;
    $\epsilon^- := \epsilon^\circ - 1;\ \eta^- := \eta^\circ - 1;\ \epsilon^+ := \epsilon^\circ + 1;\ \eta^+ := \eta^\circ + 1$;
    $d^{e-} := e - \mathcal{E}_{\epsilon^-,e};\ d^{e+} := \mathcal{E}_{\epsilon^+,e} - e;\ d^{n-} := n - \mathcal{N}_{\eta^-,n};\ d^{n+} := \mathcal{N}_{\eta^+,n} - n$;
    while $(\bar{d} \geq \min(d^{e-}, d^{e+}, d^{n-}, d^{n+}))$
      if $(\bar{d} \geq d^{e-})$
```

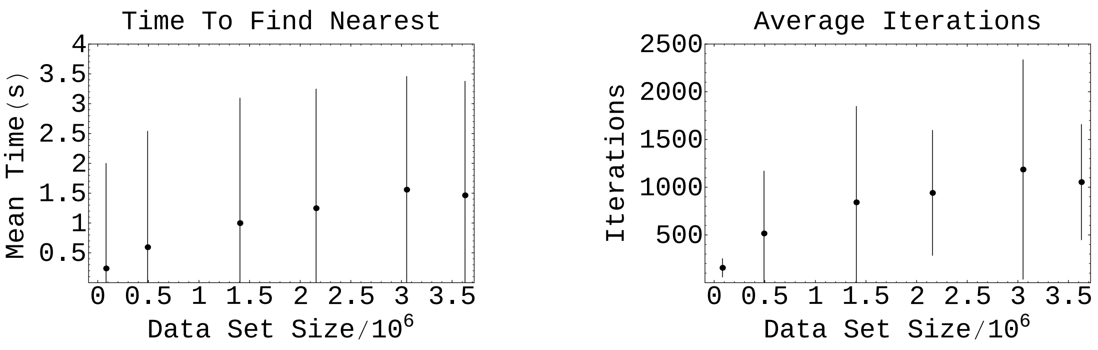

Fig. 6. (a) Execution time needed to find the posting nearest a random point. The running time is logarithmic but with great variability. (b) The average number of iterations required for a given data set size.

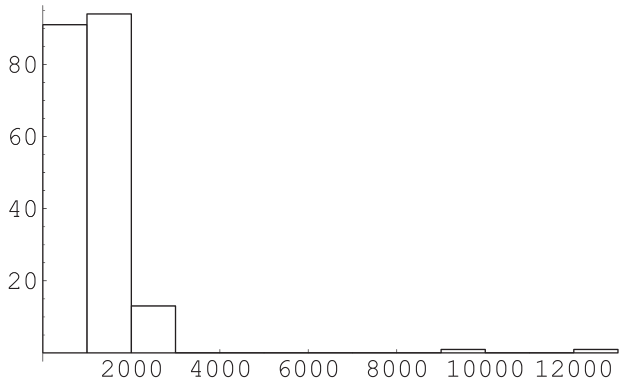

Fig. 7. Typical histogram of the iterations required to find the nearest posting. Distribution appears lognormal. The large variability indicated in Fig. 6 is due mainly to several very large, but infrequent occurrences.

```
    $\bar{d} := d^{e-}$; $\mathcal{I}_N := \mathcal{E}_{\epsilon^-,\epsilon}$; if $(\epsilon^- > 0)\{\epsilon^- = \epsilon^- - 1;\ d^{e-} := e - \mathcal{E}_{\epsilon^-,e};\}$
if $(\bar{d} \geq d^{e+})$
    $\bar{d} := d^{e+}$; $\mathcal{I}_N := \mathcal{E}_{\epsilon^+,\epsilon}$; if $(\epsilon^+ < \mathrm{N} - 1)\{\epsilon^+ = \epsilon^+ + 1;\ d^{e+} := \mathcal{E}_{\epsilon^+,e} - e;\}$
if $(\bar{d} \geq d^{n-})$
    $\bar{d} := d^{n-}$; $\mathcal{I}_N := \mathcal{N}_{\eta^-,\eta}$; if $(\eta^- > 0)\{\eta^- = \eta^- - 1;\ d^{n-} := n - \mathcal{N}_{\eta^-,n};\}$
if $(\bar{d} \geq d^{n+})$
    $\bar{d} := d^{n+}$; $\mathcal{I}_N := \mathcal{N}_{\eta^+,\eta}$; if $(\eta^+ < \mathrm{N} - 1)\{\eta^+ = \eta^+ + 1;\ d^{n+} := \mathcal{N}_{\eta^+,n} - n;\}$
```

As an example, again consider Figure 5. The circular arc section shows that posting 6 is closest to p. At the first step of the iteration, $\bar{d}$ is the distance from posting 1 to p. $\epsilon^\circ = 5$ because posting 1's easting was closest to p and e_1 is the fifth element of $\mathcal{E}$. $\eta^\circ = 2$. Therefore, start searching $\mathcal{E}$ at 4 and 6; start searching $\mathcal{N}$ at 1 and 3. Of these three postings (4, 7, 3 with 3 occurring twice), posting 4 is the closest. Therefore, $\hat{d}^*$ is set to the distance from posting 4 to p. The next iteration proceeds only increasing for $\mathcal{N}$ and decreasing for $\mathcal{E}$, having come to the end in the other directions. This iteration finds postings 6 and 5. Posting 6 is closer than 4 thus reducing $\hat{d}^*$. At the next iteration, all points are further than $\hat{d}^*$ in just their distance to p in easting or northing alone, and the algorithm terminates.

The computational complexity of this algorithm is $O(\max(\lg \mathrm{N}, \mathrm{N}'))$, where N is the number of postings and N' is the length of the larger block to search over. Although N' can be zero because the nearest point can be found without searching $\mathcal{E}$ or $\mathcal{N}$ at all, this happened extremely rarely in our testing (0.0028%). As shown in Figs 6 and 7, N' is usually small ($\mathrm{N}' \ll \mathrm{N}$) but will increase as N increases, assuming posting density is constant. In fact, in the worst case, it is possible that $\mathrm{N}' = \mathrm{N}$. More formally, suppose the postings define a square region of side length h, with statistically uniformly distributed posting density ρ, and average separation distance d. Then, the point closest to a point of interest will typically be not farther than d from the point of interest. Define two strips, one horizontal and one vertical, of width $2d$. The area of each strip is $2dh$ for a total area of $4dh$ ignoring the overlapping area. The expected number of postings in the strips is $\mathrm{N}' = 4\rho\, d\, h$. The total number of postings is $\mathrm{N} = \rho h^2$. The ratio of the number of postings in the strips to the total number of postings is $4\rho\, d\, h/\rho h^2 = 4d/h$. LIDAR postings are typically very dense so $d \ll h$ and therefore $\mathrm{N}' \ll \mathrm{N}$. Fig. 6 shows elapsed execution times for topographic data sets larger than 3.5 million postings. The graph shows the logarithmic increase with N, as expected. The large variability can be explained by Fig. 7, which illustrates that most of the iterations are fairly consistent but there are occasional very large occurrences, thus creating the large variance estimates.

3.4 Nearest Neighbors

Nearest neighbors are a set of some size of those points closest some point of interest. There are different ideas about what constitutes the nearest neighbors of a point of interest p. Alternatives include all postings inside a window of some size centered at p (Isaaks and Srivastava, 1989) or the nearest neighbors in the Delaunay sense (Gold, 1989). We also note that there is more than one way to triangulate any set of postings (e.g., see Abdelguerfi et al. (1998) or (Wang et al., 2001)) so the Delaunay definition cannot be universally agreed on. Therefore, because there is no consensus about the definition of nearest neighbors, we will define the nearest neighbors of order h to be the set $\mathcal{H}_h$ having at least h postings such that there are no other postings in $P - \mathcal{H}_h$ that are closer to p than those in $\mathcal{H}_h$. We stipulate that $\mathcal{H}_h$ has *at least* h postings so as to include multiple postings equidistant from p. For example, suppose p happened to be in the center of a circle of postings and that there were no other postings within the circle. Then, $\mathcal{H}_1$ would include all the postings on the circle. From this definition, one can observe that h is more like a ranking than the size of $\mathcal{H}$.

Finding nearest neighbors is a straightforward generalization of the $\mathcal{I}_N$ algo-

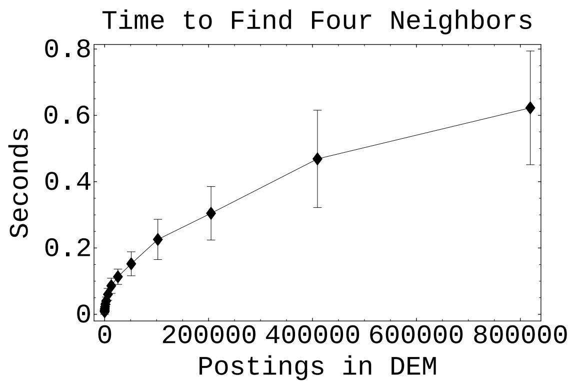

Fig. 8. Execution time needed to find the four postings nearest a random point. The ordinate is the number of postings in the DEM.

rithm. First, note that $\mathcal{I}_N$ is the set of indices of nearest neighbors of order 1 for p, that is, $\mathcal{H}_1(p) = \mathcal{I}_N(p)$. The $\mathcal{I}_N$ algorithm kept track of only the closest posting. However, if the algorithm is augmented to keep track of the h closest postings, then the result will be the nearest neighbors. The details of the program become somewhat tedious and are omitted for brevity.

The execution time of the algorithm is graphed in Fig. 8, in which neighborhoods of four postings around 20 random points of interest in DEMs of increasing size were found. In comparing Fig. 8 with Fig. 6 one sees the same logarithmic computational complexity but the time for finding four nearest points is roughly twice as long as finding only the first closest point.

3.5 SortCCW

It is often useful to order nearest neighbors' postings radially around the point of interest. The following is an algorithm that will order the postings counter-clockwise around p but without computing any trigonometric or transcendental functions. Let P_N be the postings to order around p. Let v_N be the set of vectors from p to the postings in P_N, i.e.,$v^i = p^i - p$. Now, note that within a quadrant, $\cos\theta$ is a strictly increasing function of θ so ordering by $\cos\theta$ produces the same result as ordering by θ itself. Furthermore, $\cos\theta^i = v_n^i / v_e^i$ so it suffices to sort by v_n^i / v_e^i thus eliminating the need to compute the arccosine explicitly. It remains to disambiguate by quadrant. We define a function, $q(v)$, to do this.

$$q(v) = \begin{cases} 1, \text{ if } e \geq 0 \wedge n \geq 0; \\ 2, \text{ if } e < 0 \wedge n > 0; \\ 3, \text{ if } e \leq 0 \wedge n \leq 0; \\ 4, \text{ if } e > 0 \wedge n < 0. \end{cases}$$

Then, form $S = \{\{q(v^1), v_n^1/v_e^1\}, \ldots, \{q(v^N), v_n^N/v_e^N\}\}$ and sort S first by quadrant and then by v_n^i/v_e^i within quadrant. If $v_e^i = 0$, use ∞ for v_n^i/v_e^i.

4 Discussion

Storage costs are a major concern in digital terrain modeling. Irregularly-spaced postings must have their coordinates stored explicitly although there are efficient methods to do this (Meyer, 2002). Spatial data structures, however, will typically require more storage space than the topographic data set itself. The approach given in this article requires external storage $O(N)$, where N is the number of postings, whereas quadtrees, k-d-B-trees, hB-trees and R-trees require storage $O(N \log N)$. These access method partition space hierarchically into regions that may or may not overlap. Queries are answered by traversing the tree to identify regions satisfying the query criteria; the hierarchy enabling logarithmic search complexity. Range searching views spatial queries to be predications of the points in the intersection of sets of half-spaces (Arge et al., 1999), a perspective that arose from constraint database theory. Range searching is supported by sorted arrays (Arge et al., 1999) in conjunction with weighted B-trees (Arge and Vitter, 1996), priority search trees (McCreight, 1985) or p-range trees (Subramanian and Ramaswamy, 1995) and, consequently, require storage $O(N \log N)$. A Delaunay triangulation, represented as a list of nearest-neighbor lists, is also linear. However, an efficient implementation requires a hash table or associative array to store the variable length nearest-neighbor lists, which on average, have six edges between postings for every posting in the topographic data set. In contrast, the method in this paper adds exactly three indices per posting and incurs no overhead for a hash table or associative array. The storage overhead for this method is low. However, the aforementioned tree-based methods readily support updates, which the proposed method does not. This decision is acceptable in practice because large topographic data sets tend to be static. Once a data vendor has created and edited a data set, insertions and deletions seldom occur. This is typically true for data, as well. Subsets of a dataset might be extracted for specific purposes but data are typically not added or deleted from the original dataset piecemeal.

Constructing an access method can, itself, be prohibitively time-consuming. Our method requires three sortings of the data and is, therefore, $O(N \log N)$. Algorithms of this complexity exist for the other spatial access methods (Delaunay triangulations, R-trees, etc.), too. However, our data structure is very simple and the constant of proportionality for its construction is small. For a specific example, constructing a Delaunay triangulation of only 50 000 in-memory postings in *Mathematica* v 5.1 took more than 95 minutes. Our implementation, using Unix operating system sorting and cutting operations, builds the data structures in less than 12 minutes for a 3.6 million posting data set. Although there are obvious "apples to oranges" comparison problems, the example illustrates that the more complicated algorithms can run prohibitively slowly. This was a primary motivation for the current investigation, in fact.

Finding the posting nearest some point of interest requires time $O(\sqrt{N})$ for a Delaunay triangulation and $O(\log N)$ for tree-based methods; the algorithm in this paper is $O(\max \lg N, N')$, where N' is the number of postings whose individual easting or northing coordinate is closer to the respective coordinate of the point of interest than the corresponding coordinate of the closest posting. Finding the nearest neighbors of some point of interest requires constant time for a Delaunay triangulation and $O(\log N)$ for tree-based methods; the algorithm in this paper is comparable. Thus, the data structures presented in this paper require potentially far less external storage than the alternatives and are computed very quickly, and the searching algorithms generally either run faster or comparably. These characteristics suggest this approach to be well-suited for large sets of topographic postings while maintaining the advantages of irregular spacing. Although these algorithms were written with an eye towards digital terrain modeling, they can be generalized to higher dimensional datasets by adding more sorted arrays equivalent to $\mathcal{E}$ and $\mathcal{N}$ for the higher dimensions.

5 Acknowledgments

The author gratefully acknowledges the comments and suggestions of three anonymous reviewers, whose input improved the scope and depth of this investigation. Sandy Prisloe generously provided the LIDAR data set which was acquired through a coastal remote sensing project supported by the National Oceanic & Atmospheric Administration's Coastal Services Center.

References

Abdelguerfi, M., Wynne, C., Cooper, E., Ladner, R., 1998. Representation of 3-d elevation in terrain databases using hierarchical triangulated irregular networks: a comparative analysis. International Journal of Geographical Information Science 12 (8), 853–873.

Arge, L., Samoladas, V., Vitter, J. S., 1999. On two-dimensional indexability and optimal range search indexing. In: Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. ACM Press, Philadelphia, Pennsylvania, pp. 346–357.

Arge, L., Vitter, J. S., 1996. Optimal dynamic interval management in external memory. In: Proceedings of the IEEE Symposium on Foundations of Computer Science. pp. 560–569.

Baltsavias, E., 1999. Airborne laser scanning: basic relations and formulas. ISPRS Journal of Photogrammetry & Remote Sensing 54 (2-3), 199–214.

Bentley, J., 1975. Multidimensional binary search used for associative searching. Communications of the ACM 18 (9), 509–517.

Cheng, Q., 1999. The gliding box method for multifractal modeling. Computers & Geosciences 25 (9), 1073–1079.

Cooper, G., Cowan, D., 2004. The detection of circular features in irregularly spaced data. Computers & Geosciences 30 (1), 101–105.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., 1997. Introduction to Algorithms. McGraw-Hill Book Company, New York, p. 1028.

de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., 1998. Computational Geometry: Algorithms and Applications. Springer–Verlag, New York, p. 367.

DeCola, L., 1989. Fractal analysis of a classified LANDSAT scene. Photogrammetric Engineering and Remote Sensing 55 (5), 601–610.

Douglas, D. H., 1986. Experiments to locate ridges and channels to create a new type of digital elevation model. Cartographica 23 (4), 29–61.

Flood, M., Gutelius, B., Apr. 1997. Commercial implications of topographic terrain mapping using scanning airborne laser radar. Photogrammetric Engineering & Remote Sensing LXIII (4), 327–329, 363–366.

Gamba, P., Houshmand, B., 2000. Digital surface models and building extraction: A comparison of IFSAR and LIDAR data. IEEE Transactions on Geoscience and Remote Sensing 38 (4), 1959–1968.

Gebhardt, F., 2001. Spatial cluster test based on triplets of districts. Computers & Geosciences 27 (3), 279–288.

Gold, C. M., 1989. Surface interpolation, spatial adjacency and GIS. In: Raper, J. (Ed.), Three Dimensional Applications in Geographical Information Systems. Taylor & Francis, pp. 21–35.

Gould, P., 1981. Letting the data speak for themselves. Annals of the Association of American Geographers 71 (2), 166–176.

Guttman, A., 1984. R-trees: a dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD International Conference on the Management of Data. Boston, pp. 47–57.

Hessami, M., Anctil, F., Viau, A. A., 2001. Delaunay implementation to improve kriging computation efficiency. Computers & Geosciences 27 (2), 237–240.

Hodgson, Michael, E., Jensen, J. R., Schmidt, L., Schill, S., Davis, B., 2003. An evaluation of LIDAR- and IFSAR-derived digital elevation models in leaf-on conditions with USGS Level 1 and Level 2 DEMs. Remote Sensing of the Environment 84 (2), 295–308.

Isaaks, E. H., Srivastava, R. M., 1989. An Introduction to Applied Geostatistics. Oxford University Press, New York, p. 561.

Lomet, D., Salzberg, B., 1989. A robust multi-attribute search structure. In: Proceedings of the 5th International Conference on Data Engineering. Los Angeles, California, pp. 296–304.

Lomet, D., Salzberg, B., 1990. The hB-tree: A multiattribute indexing method with good guaranteed performance. ACM Transactions on Database Systems 15 (4), 625–658.

Makarovic, B., 1977. From progressive sampling to composite sampling for digital terrain models. Geo–Processing 1, 145–166.

McCreight, E., 1985. Priority search trees. SIAM Journal of Computing 14 (2), 257–276.

Mercer, J. B., Schnick, S., Nov. 1999. Comparison of DEMs from STAR-3i(r) interferometric sar and scanning laser. In: Proceedings of the ISPRS WG III/2,5 Workshop: Mapping Surface Structure and Topography by Airborne and Spaceborne Lasers. La Jolla, California, pp. 127–134.

Meyer, T. H., 2002. A near-optimal data structure for storing, retrieving, and indexing three-dimensional geospatial data. In: Proceedings of FIGS/ACSM/ASPRS Annual Convention. pp. on CD–ROM.

Meyer, T. H., Eriksson, M., Maggio, R. C., 2001. Gradient estimation from irregularly spaced data sets. Mathematical Geology 33 (6), 693–717.

Mortenson, M. E., 1985. Geometric Modeling. John Wiley & Sons, New York, p. 764.

Nievergelt, J., Widmayer, P., 1991. Spatial data structures: concepts and design choices. Lecture Notes in Computer Science. Springer, New York, pp. 153–197.

Optech, 2003. Want more choice? Optech's ALTM gives you 70,000 Hz. http://www.optech.on.ca/pdf/ALTM3070.pdf.

Ozkaya, S. I., 2002. QUADRO - a program to estimate principal curvatures of folds. Computers and Geosciences 28 (4), 467–472.

Philip, G. M., Watson, D. F., 1986. A method for assessing local variation among scattered measurements. Mathematical Geology 18 (8), 759–764.

Robinson, J., 1981. The K-D-B-tree: a search structure for large multidimensional dynamic indexes. In: Proceedings of the ACM SIGMOD International Conference on the Management of Data. Ann Arbor, pp. 10–18.

Samet, H., 1990. Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley Publishing Company, Reading, Massachusetts, p. 507.

Shary, P. A., 1995. Land surface in gravity points classification by a complete system of curvatures. Mathematical Geology 27 (3), 373–390.

Subramanian, S., Ramaswamy, S., 1995. The P-range tree: a new data structure for range searching in secondard memory. In: Proceedings ACM-SIAM Symposium on Discrete Algorithms. pp. 378–387.

Wang, K., Chor-Pang, L., Brook, G. A., Arabnia, H. R., 2001. Comparison of existing triangulation methods for regularly and irregularly spaced height fields. International Journal of Geographic Information Science 15 (8), 743–762.