Transactions in GIS, 2000, 4(4): 379–408

Research Article

Multiscale Terrain and Topographic Modelling with the Implicit TIN

David B Kidner
School of Computing
University of Glamorgan

J. Mark Ware
School of Computing
University of Glamorgan

Andrew J Sparkes
School of Computing
University of Glamorgan

Christopher B Jones
Department of Computer Science
Cardiff University

Abstract
The Multiscale Implicit Triangulated Irregular Network (TIN) provides a storage and access scheme for generating triangulated terrain models that adapt their content and level of detail to the requirements of the user. The scheme combines storage of data representing the terrain surface, and two and three dimensional terrain features, with a retrieval and triangulation procedure that generates a constrained Delaunay triangulation at run time. The feature content and level of detail may be specified by the user, thereby providing a flexible facility that adapts to the requirements of a wide range of applications, whether global or local, exploratory or precise. This paper provides an overview of the scheme and illustrates its application for a variety of queries requiring multiscale representations.

1 Introduction

A digital terrain model (DTM) can be described generically as a digital representation of a portion of the Earth's surface. There are a number of approaches for digitally representing the elevation component of a surface, which are often governed by the method of data acquisition. In essence, elevations can be represented by regularly sampled grids in the x–y plane (termed digital elevation models or DEMs), lines of constant elevation or contours, or at irregularly sampled points or lines which best characterise the terrain (e.g. spot heights and breaklines). While there is a proliferation

Address for correspondence: David B Kidner, School of Computing, University of Glamorgan, Pontypridd, Rhondda Cynon Taff, Wales CF37 1DL. E-mail: dbk.dner@glam.ac.uk.

of DEMs in the public domain, there are many more surfaces represented as contours and other terrain-specific features. However, digital representations of contours and irregularly sampled elevations are less well understood and often dismissed by GIS users who prefer the quick-fix solution of the DEM. The choice of DTM has been the subject of debate in the GIS community for the last 25 years, while comparative studies are often inconclusive (e.g. Kidner 1991, Kumler 1994). This is not too surprising, as independent sources of elevation data for different DTMs are rare. As a consequence, different terrain models are derived from each other rather than directly from the original surface, resulting in a bias in any comparisons. For example, irregularly sampled elevation data may be interpolated onto a regular grid DEM (Clarke et al. 1982, Mitas and Mitasova 1999) and then a triangulated irregular network (TIN) may be derived from the DEM (Lee 1991a).

The TIN approximates a surface using a network of planar, non-overlapping, and irregularly shaped triangle faces (De Floriani 1987). The emphasis is on modelling the terrain surface, not the elevations. As such, characteristics of the surface such as breaklines can be maintained within the TIN, in addition to contours and spot heights. The arguments in favour of the TIN are well-founded (i.e. adaptive to the variability of the terrain, overcomes the data redundancy of the DEM, etc.), while critics of the DEM argue that the grid cell is an aberration which over-simplifies terrain modelling (Lee 1991b). In comparison, the triangle cell of the TIN should best represent the surface behaviour between elevation samples. This is essential for modelling many spatial processes of the physical environment that are dependent upon terrain, such as landslides, hydrology, or erosion (Weibel 1997).

Rather than extend the DEM versus TIN debate further, this paper will attempt to demonstrate the virtues of the TIN for *flexible* terrain modelling. As digital terrain modelling becomes more widespread, particularly in a desktop GIS environment (e.g. 3D Analyst, Vertical Mapper), users are demanding more advanced functionality and a range of modelling scenarios. For example, the opportunity to integrate the DTM with digital representations of topographic features on the surface. In this context, the DTM is often referred to as a digital surface model (DSM). A DSM represents the top of the terrain cover, such as the tree canopy and roofs of buildings, rather than the ground elevations themselves (Weibel 1997). This usage of DSM is particularly widespread in conjunction with automated remote sensing techniques for terrain data capture which can measure this kind of surface, including digital photogrammetry, laser scanning or radar interferometry (Weibel 1997). It is now widely accepted that TINs can be extended to accommodate surface features on the landscape. The paper addresses this issue and illustrates how surface features can be inserted into a TIN.

This paper primarily focuses on a data access scheme for digital terrain modelling that offers users the flexibility to handle multiscale queries. The *Implicit TIN* is a DTM that gives users a *fitness-for-use* solution to the problem in hand. As such, it is not constrained to a single scale representation of terrain or its surface features, but acknowledges that user queries are invariably unique, hypothetical and exploratory in nature. A range of modelling strategies is often best suited for these queries, for example, at small scale or large scale and with or without topographic surface features. Section 2 introduces the dilemma faced by digital terrain modellers that have opted for the triangulated irregular network (TIN) – that of the choice of data structure for maintaining the topology of the TIN. Section 3 presents the reasoning behind the *Implicit TIN* that helps to solve this dilemma for users. The process of creating a user-

specific TIN is then described. Section 4 extends this discussion by considering how the *Implicit TIN* can maintain multiscale representations of surface, contour, and topographic data. The problem of representing and inserting topographic surface features into the TIN is presented in Section 5. A range of multiscale modelling queries are then illustrated in Section 6 for intervisibility analysis, range-based queries, and level-of-detail (LOD) representations. The paper concludes with some recommendations for the way forward.

2 Data Structures for TINs

Data storage schemes based on conventional TIN data structures incur a storage overhead due to the fact that they represent the topology of the triangulation explicitly. The spatial relationships maintained by this topology determine both the computational efficiency and storage efficiency of the TIN. In general, a TIN which utilises a fraction of the number of vertices of its equivalent regular grid digital elevation model (DEM) will often require a higher storage or memory requirement than the latter.

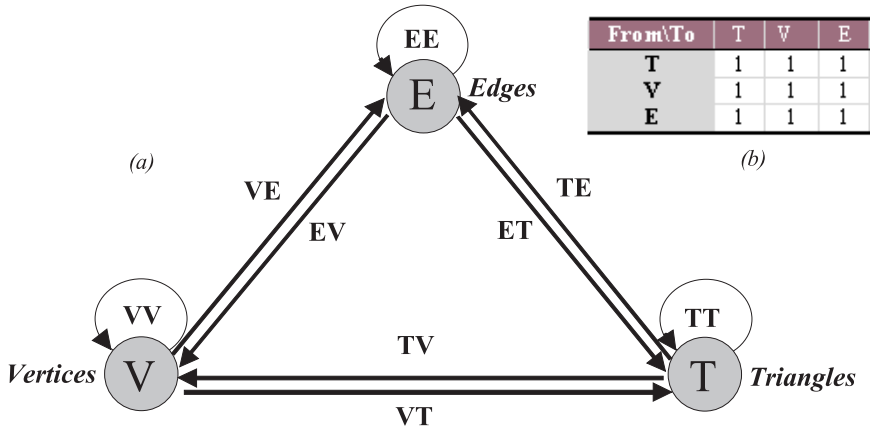
In creating databases that represent TIN-based data structures, a large amount of storage is taken up by the pointers used to maintain the topological relationships between entities, such as the connectivity of triangle vertices and edges. Hence, the data structure of any TIN can be represented in a variety of different ways. The primary entities of each structure represent the three primitive entities of a TIN – vertices, triangles and edges. De Floriani (1987) states that the topology of a triangular subdivision is completely and unambiguously represented by any suitably selected subset of nine possible adjacency relations between these entities. These are illustrated in Figure 1a, using an arrow diagram, where each arrow indicates an ordered relation between the pair that may be represented in the data structure. Figure 1b illustrates these equivalent relations as a matrix representation.

An illustration detailing these relations is shown in Figure 2. The choice of which entity relation is used depends largely upon the application for which the TIN is intended. Each TIN data structure has evolved through the requirements of specific applications, such that they each have their own distinct advantages and disadvantages (Kidner and Jones 1991).

For example, consider the TIN data model that represents the triangles as the primary entity. Each triangle is uniquely referenced and is defined by pointers to its three defining vertices and its neighbouring triangles. The coordinates of all the vertices must still be stored in a secondary file. The entity relations required to define this model can be derived from the nine possible topological relationships as *triangle-to-triangle* (TT) and *triangle-to-vertex* (TV) (Figure 3).

In relation to the example presented in Figure 2, the actual data structure as a triangle-based TIN is presented in Table 1, where the Null triangle is defined as the region external to the area defined. In this example, the order of the adjacent triangles is derived from the corresponding order of the clockwise vertices.

However, the original TIN presented by Peucker et al. (1978) proposed a data structure in which the primary entities are the vertices themselves. In essence, the list of neighbouring vertices define the edges of the TIN, while the triangles can themselves be implicitly inferred from these. Alternatively, Heller (1990) advocates the use of an edge-



From\To	T	V	E
T	1	1	1
V	1	1	1
E	1	1	1

(b)

From\To	T	V	E
T	1	1	0
V	0	0	0
E	0	0	0

Figure 1 Topological relations in a plane (triangular) subdivision (after Woo 1985) (a) as an arrow graph (b) as a matrix

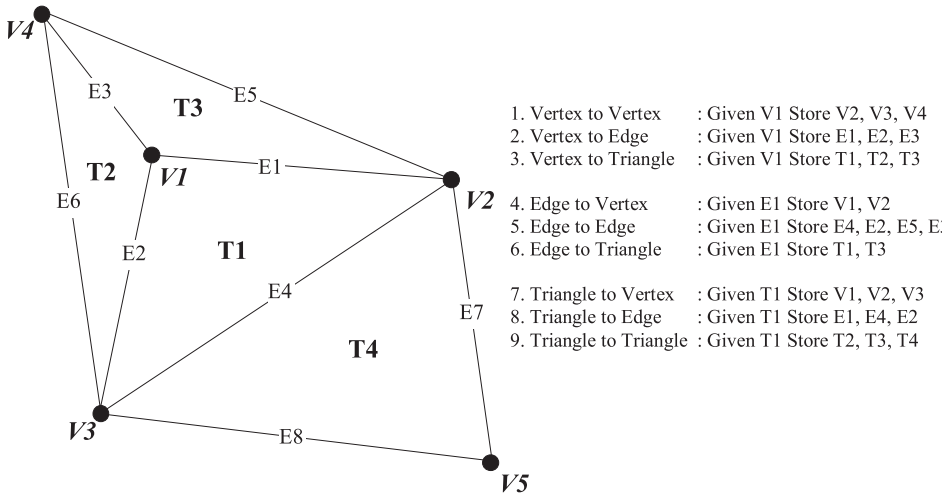
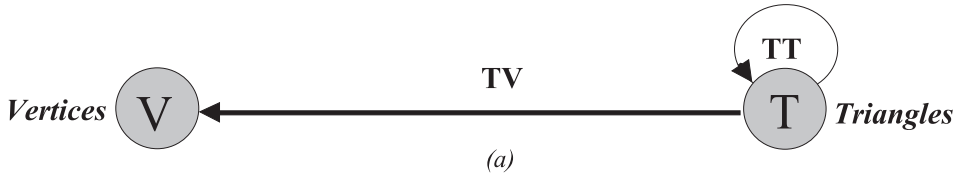


Figure 2 Illustration of the nine possible topological relations between pairs of entities in a TIN (where Vn represent vertices; En represent edges; and Tn represent triangles)

orientated data structure, as it is better suited for swapping and splicing of triangle edges, which is particularly beneficial for dynamic triangulation.

Each data structure is better suited for particular types of operations. For example, if the TIN is to be used for the interpolation of points or the derivation of contours,



(a)

From\To	T	V	E
T	1	1	0
V	0	0	0
E	0	0	0

(b)

Figure 3 The two topological relationships required to define a TIN where the primary entities are the triangles themselves (a) as a graph (b) as a matrix

Table 1 Triangle-based TIN for the example of Figure 2

Triangles	Vertices (i.e. TV Relation)	Adjacent Triangles (i.e. TT Relation)
T1	V1, V2, V3	T3, T4, T2
T2	V4, V1, V3	T3, T1, Null
T3	V1, V4, V2	T2, Null, T1
T4	V3, V2, V5	T1, Null, Null

then a triangle-based data structure may be preferable. If the TIN is to be used for visibility analysis, in which lines-of-sight are calculated by interpolating through the triangle sides, then an edge-based data structure can be considered more efficient. However, in order to minimise the data storage and hence memory requirements, then a vertex-based data structure is better, but possibly at a cost of increased computation to infer the implicit topological relationships. In essence, no data structure is clearly superior for all tasks of terrain modelling.

3 The Implicit TIN

The *Implicit TIN* differs from a conventional TIN in that only the vertices are explicitly stored, together with the definition of any linear constraints. No topological relationships defining the triangulation are recorded. TIN topology is reconstructed by a triangulation procedure if and when it is required for a user operation. The triangulation procedure can itself be thought of as being an integral component of the *Implicit TIN*. However, the advantage of a highly compact storage scheme must be weighed against the cost of having to reconstruct the TIN topology when required. In practice though, this is a small price to pay for the benefits and flexibility that the *Implicit TIN* can deliver.

The *Implicit TIN* offers a number of advantages over the conventional explicit TIN data structures. In the first instance, the topological relations and hence the data structure that is derived from the *Implicit TIN* can be *made-to-order* for the intended application of the user. This can help to offset the increased time required for retrieving the necessary topology. Secondly, as the *Implicit TIN* can be thought of as a set of raw vertices, more refined search algorithms can be utilised to only retrieve the spatial data pertinent to any user query. As such, the reconstructed TIN might only require a fraction of the main memory requirement of a conventional TIN, thus significantly improving real-time search operations. Kidner and Jones (1991) and Jones et al. (1994) demonstrate how small subsets of the complete data set can be retrieved to generate more computationally efficient TINs. Thirdly, as digital terrain models are now more likely to be nationally available at multiple scales or resolutions, the *Implicit TIN* is well placed to capitalise on these developments as it supports multiscale vertices. The user specifies a query scale, such that only those vertices up to that scale tolerance are retrieved for generating the TIN. As such, an operation such as an exploratory viewshed calculation can be generated at a small or medium scale, which if satisfactory, can then be repeated at the largest scale, with all TINs retrieved from the one data model. The *Implicit TIN* also supports the retrieval of vertices at multiple scales for range-based queries or multiple level-of-detail (LOD) visualisations. Finally, the *Implicit TIN* supports the integration of topographic features, such as buildings or vegetation, if the user requires it. For most conventional TINs, topographic features are often ignored, to the detriment of many applications, such as visibility analysis. The storage overheads of maintaining complex objects in a TIN can be quite significant, even for small geographical areas, but the *Implicit TIN* gives users the option to include or exclude such features, again for more manageable user-defined query windows.

The basic components of the *Implicit TIN* are illustrated in Figure 4. Source data consists of point data, held in the *Point File*, and edge data, held in the *Edge File*. The *Point File* stores details relating to a collection of irregularly distributed elevation points, and can be thought of as the primary source of data for the TIN. Each record stores a unique identifier and x, y, z coordinate for a single point (vertex) location. The *Edge File* stores data relating to surface constraining linear features such as a shoreline or a building in the form of a series of edges. In practice, there could be an *Edge File* and *Edge Quadtree* for distinct geographical themes, e.g. contours, roads, buildings, etc. Each record consists of a unique identifier and either the x, y, z coordinates of the two defining edge vertices or their *Point File* IDs. In the case of the base elevations of the constraining features being unknown, such as for a building footprint, they are interpolated from the underlying terrain using the ray-shooting algorithms described by Clarke et al. (1982).

Two spatial indexes, the *Point Quadtree* and the *Edge Quadtree* are employed during construction of the TIN. These quadtrees are closely related to the PMR-quadtree in structure (Nelson and Samet 1986). Each *Point Quadtree* cell references all the elevation vertices that lie within it, while each *Edge Quadtree* cell references all edges that intersect it. The TIN construction algorithm constructs a constrained Delaunay triangulation for a given, arbitrarily shaped, query window. It does this in four main steps:

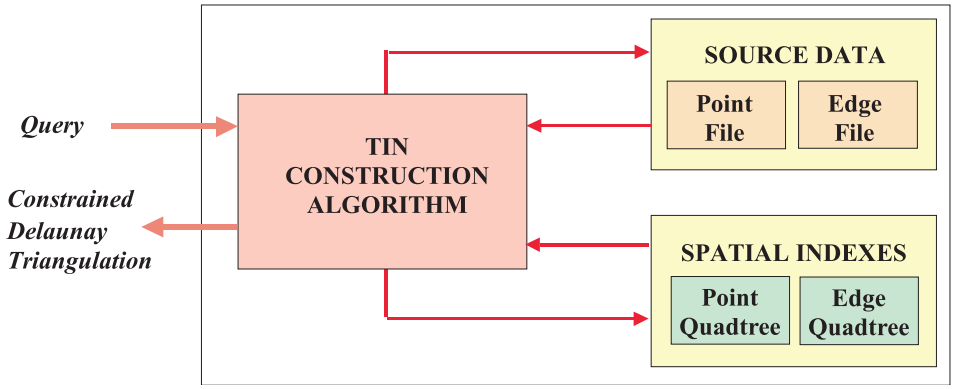


Figure 4 The framework for defining a constrained Implicit TIN

Initial Data Retrieval

Initially, lists of *Point Quadtree* and *Edge Quadtree* cells that intersect the query window are generated. All point and edge data belonging to the required classes are then retrieved. Any additional data that may be required to ensure consistency are retrieved during triangulation. The retrieved elevation vertices and edge constraints are then stored in a main-memory regular grid structure that provides additional spatial indexing during triangulation.

Initial Triangulation

The second stage involves the construction of the Delaunay triangulation of all relevant points, from both elevation data and edges. Points that lie inside the query region will always belong to the final triangulation, so initially these points are placed on a stack of points to be triangulated. The Thiessen neighbours of each point P on the stack are then found in the following way. The nearest neighbour N of P is regarded as the first Thiessen neighbour. The next Thiessen neighbour K, to the right of the edge P-N, a Delaunay edge, is then found and added to the list of Thiessen neighbours. The Thiessen neighbour to the right of the edge P-K, also a Delaunay edge, is then found. The process is repeated until the latest neighbour K is equal to the original neighbour N. The search for Thiessen neighbours makes use of the regular grid indexing structure, such that only points within the vicinity of a Delaunay edge are tested. If the search for a Thiessen neighbour includes regular grid cells which are empty (that is, lie outside the generated *Point Quadtree* region) or extends beyond the regular grid coverage, the required *Point Quadtree* cells intersected by the local search region are accessed and the necessary vertex information is retrieved. Similarly, if the search extends beyond the previously generated *Edge Quadtree* region, then the *Edge Quadtree* cells intersected by the local search region are accessed and edge information is retrieved. When all Thiessen neighbours of P have been found, the point is removed from the stack and added, together with its list of neighbours, to a list of processed points. This list, when complete, can be used to formulate a vertex, edge, or triangle primary entity TIN. The Thiessen neighbours of the next point on the stack are found in like fashion. The

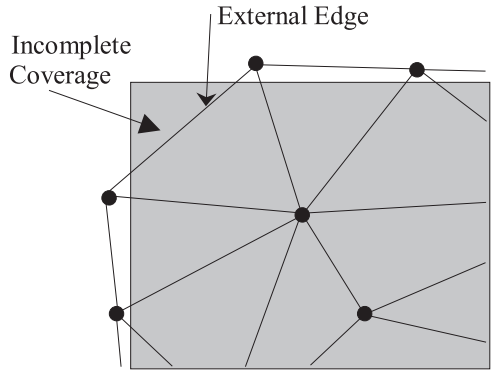


Figure 5 Initial triangulation does not guarantee complete coverage

process is repeated until the stack is emptied. The algorithm is largely based on the approach of McCullagh and Ross (1980), who describe a simple approach for triangulating irregular data with a grid cell indexing scheme.

Complete Coverage Check

A triangulation of all points lying inside the query region will not guarantee a complete TIN coverage over that region. There is the possibility of parts of the query window not being covered, especially in its corners where a Delaunay edge might cross the window but both its vertices are outside (Figure 5). The algorithm caters for such occurrences by checking for such edges, termed external edges, and when found, adding the points belonging to the edges to the stack of vertices to be triangulated. Thus when the triangulation is complete, triangles will have been constructed on both sides of all such edges.

Insertion of Constraints

The final stage in the triangulation process is to insert the constraints conforming to all *Edge File* line segments which lie within or intersect the current TIN. Each constraining edge (*p1*, *p2*) can have one of four possible states: (1) *p1* and *p2* are both vertices within the TIN and form a Delaunay edge; (2) *p1* and *p2* are both vertices within the TIN but are not connected; (3) either *p1* or *p2* is a TIN vertex while the other is external to the TIN; or (4) both *p1* and *p2* are external to the TIN. In the first case (1), no update is necessary. In the other cases the segment does not exist and therefore the TIN must be constrained. For case (2), the procedure for inserting the edge can be broken down into four steps. The first consists of determining the current edges that are intersected by the constraint. Step two deletes these edges, while the third step involves re-triangulating around the new edge. Finally, step four deals with updating the TIN data structure.

The problem of re-triangulating around the constraining edge is reduced to that of separately triangulating the two polygons formed either side of the edge. These polygons are sometimes referred to as the polygons of influence (De Floriani and

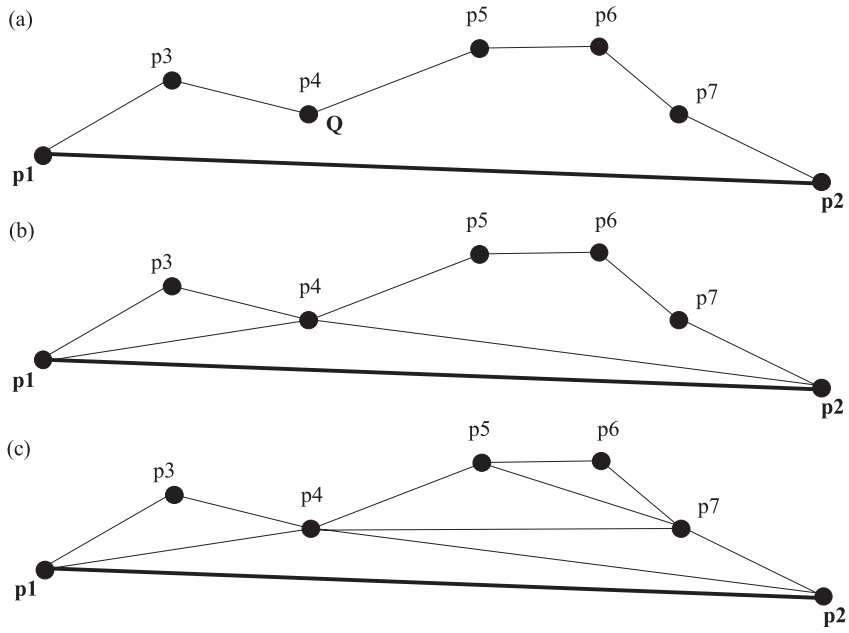


Figure 6 Triangulation of a polygon of influence

Puppo 1988). The triangulation of each polygon proceeds as follows. Consider the edge (*p1*, *p2*) to be the base edge of the polygon. The initial step is to find the vertex *Q* of the polygon, discounting the vertices *p1* and *p2*, which subtends the largest angle to the base edge. For the polygon shown in Figure 6a this is vertex *p4*. This vertex is added to the list of neighbours of both *p1* and *p2*, and similarly *p1* and *p2* become neighbours of vertex *p4*. Two sub-polygons have now been formed with base edge (*p1*, *p4*) and (*p4*, *p2*) respectively (Figure 6b). The two sub-polygons, and any subsequent sub-polygons, are dealt with, recursively, in the same way as the original polygon (Figure 6c). The recursion continues along a particular path until the latest new edge matches an edge in the original TIN. For cases (3) and (4), the procedure is similar, but the polygons of influence may now include vertices outside the original TIN.

If the raw elevation data consists predominantly of contour data, then the triangulation can result in many undesirable artefacts, particularly if a Delaunay triangulation algorithm is adopted. These include the creation of flat regions (one or more triangles formed from the same contour) and triangle edges intersecting contour lines. The latter problem is identified and remedied in the algorithm presented above. In the case of flat triangles, a number of special corrective measures are applied, such as the insertion of new points into the triangulation and the swapping of triangle edges. These issues have been addressed by a number of researchers, but the approach adopted here is based upon a modification to Ware's (1998) algorithm, specifically designed for triangulating contour data. Sparkes (2000) gives a more detailed overview of these issues and remedies, and also reports on the effect of flat triangles in a TIN used for intervisibility analysis.

4 Multiscale TINs

For many terrain modelling applications, it is useful to be able to query or visualise the surface at different levels of detail or multiple scales. Intervisibility analysis and flight simulation are two prime examples. In the former case, a number of different planning scenarios are usually examined in order to identify the optimal location for features such as missile defence sites, fire watchtowers, radiocommunications transmitters, or wind farms. Much of the preliminary analysis can be performed at a lower resolution, such that solutions or viewsheds can be displayed almost immediately. Further, more detailed analysis at the highest resolution can then be performed on those sites that meet the initial criteria. It is not unreasonable to expect some detailed viewshed computations which integrate topographic features over a large zone of visual influence on a TIN to take upwards of an hour. With respect to flight simulation, a high level of detail is needed when flying low, whereas a low level of detail is required when flying high. More precisely, there is a need to visualise the part of the scene that is close at a high level of detail, and the part that is far away at a low level of detail (de Berg and Dobrindt 1995). The key to real-time rendering of large-scale surfaces is to locally adapt the complexity of the surface geometry to changing view parameters (Hoppe 1998). Several schemes have been developed to address this problem of view-dependent level-of-detail control. Among these, the view-dependent progressive mesh (VDPM) framework represents an arbitrary triangle mesh as a hierarchy of geometrically optimised refinement transformations, from which accurate approximating meshes can be efficiently retrieved (Hoppe 1998). Certainly the focus to date on multiscale terrain modelling has been on visualisation, where the main issue is real-time rendering of terrain surfaces. However, Kofler et al. (1998) have taken this discussion further by suggesting a framework for three-dimensional modelling and data management within GIS. This takes the mesh or TIN further by spatially indexing it within an R-tree data structure.

To date, the majority of the multiscale data structures used for terrain modelling are synonomous with the progressive meshes favoured by the computer graphics industry as a means of providing real-time, level-of-detail rendering of surfaces, particularly over the internet (Lindstrom et al. 1996, Hoppe 1996, 1997, Reddy et al. 1999). It should be remembered though, that these tools are designed for real-time visualisation and not typical GIS analysis or interpolation. Moore et al. (1999) consider the various approaches for terrain visualisation in VRML and give a good overview of the use of the *IndexedFaceSet* and *ElevationGrid* primitives for TIN and DEM modelling respectively.

Access to spatial data at variable scales can be achieved by storing multiple representations of the data at predetermined scales; storing a single large-scale version from which smaller scales are derived using generalisation algorithms; or using a multiresolution data structure specifically adapted to retrieving data at varying degrees of detail. Storage of multiple representations results in significant storage overheads owing to data duplication between different versions. Retrieval from a single version can incur major processing overheads when deriving a representation of much smaller scale. Multiresolution or multiscale data structures represent a compromise between these approaches. Essentially, multiscale data models can take the form of a true *Hierarchical TIN*, in which finer resolution data is progressively nested at lower levels, or those that represent scale-dependent vertices

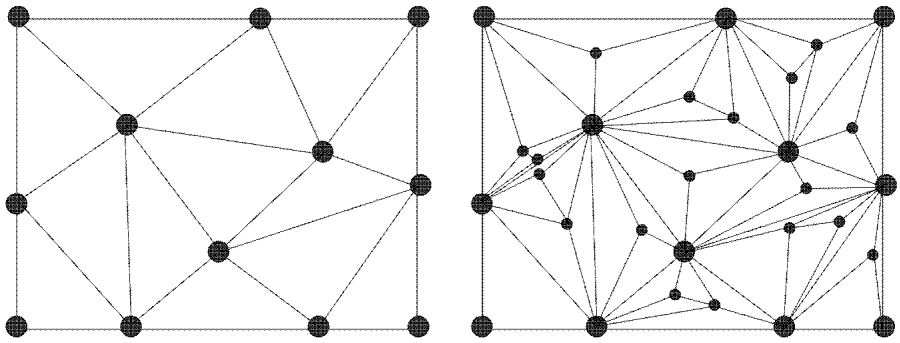


Figure 7 A two-level TIN hierarchy (with its inherent slivers)

at different levels which will require a re-triangulation between levels, such as with the Delaunay pyramid (De Floriani, 1989). Heckbert and Garland (1997) review all the main approaches used for generalising or simplifying surfaces into different levels or hierarchies and while this review does not evaluate the individual approaches, it raises an interesting question as to the arbitrariness of the different subdivision methods.

In the *Hierarchical TIN*, the initial representation is refined by adding new vertices and locally re-triangulating within each triangle. This process is repeated until all data points have been added, or some precision criterion is met. However, the edges of the initial triangulation remain present at more detailed levels (de Berg and Dobrindt, 1995). Thus any subsequent vertex inserted in close proximity to an existing edge will generate a sliver triangle (Figure 7).

Abdelguerfi et al. (1998) reviewed the aesthetic appearance of hierarchical TINs with respect to the properties of adjacency, nesting, streakiness, and sliveriness. This criterion is open to question, as slivers (i.e. very thin triangles with negligible area) can make a TIN almost unusable due to the double precision arithmetic required to minimise errors during interpolation. This is particularly evident during intervisibility analysis, as it may become almost impossible to accurately determine which triangles the terrain cross-section passes through. For all but the simplest data sets, slivers are a major concern in *Hierarchical TINs*, yet the issue has yet to be addressed with a satisfactory solution. The problem is usually minimised by the proponents of *Hierarchical TINs* by illustrating their use on data derived from low to medium resolution regular grid DEMs (cf. Abdelguerfi et al. 1998). However, several researchers have shown that slivers can sometimes be good when the surface being approximated is highly curved in one direction, but not the other (Rippa 1992). Nevertheless, the big advantage of the Hierarchical TIN is for near real-time surface visualisation, due to the simplicity by which multi-resolution scenes can be dynamically generated.

The general alternative approach to generating a multiscale TIN utilises a Delaunay triangulation of the data points at every level, which will minimise the occurrence of slivers. As the circumscribing circle of a Delaunay triangle contains no other vertices, the resultant triangulation maximises the minimum angle of all triangles, thus producing *fatter*, more equiangular triangles (Figure 8).

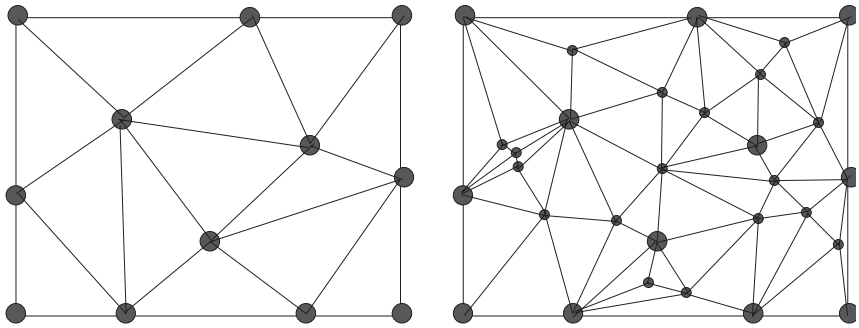


Figure 8 A two-level TIN representation based on a Delaunay triangulation

By incrementally inserting the surface elevations with the maximal error and re-triangulating, a series of Delaunay triangulations at prescribed error tolerances can be easily generated. Such a series is known as a Delaunay pyramid (De Floriani 1989) or a Pyramidal Terrain Model (PTM) (De Floriani 1995). However, the new triangles generated at any level will not be neatly nested inside a parent triangle, as with the *Hierarchical TIN*. Bertolotto et al. (1994) proposed a PTM based on a *sequence of lists of triangles* (SLT) which stores minimal topological information about entities belonging to the same level, but encodes links between entities belonging to different levels to express their spatial interference. While a formal definition of the relationship between triangles at different levels is necessary for efficient real-time applications such as flight simulation, the maintenance of the TIN topology at each level, in addition to the hierarchy topology, can be considered a storage-expensive luxury. The resolution of today's generation of TINs, derived from remotely-sensed scanners (e.g. RADAR interferometry and LiDAR laser scanning) require more efficient strategies for maintaining data. For example, a 2×2 km section of terrain sampled using a laser-scanning approach will typically generate in excess of one million x, y, z tuples. While there may be significant redundancy in these samples depending on the nature of the terrain, it is reasonable to assume that most surfaces with a sufficient geographical extent for visibility, hydrological or radiocommunication modelling will require millions of vertices. It is also reasonable to assume that not many computer systems will be capable of handling the resultant TIN in main memory, while disk-based access can be very inefficient for intensive applications such as viewshed analysis. While multiscale TINs reduce these problems for most users and applications at small to medium scales, the problems remain for those users who consistently work on large scale analysis. The key to solving this problem is to efficiently retrieve data into main memory, while minimising the geographical domain of the TIN. This can be achieved with good spatial indexing and adapting the search to the specific query or problem of the user. For example, Kofler et al. (1998) demonstrate the use of a multiscale TIN indexed by R-trees to retrieve all objects within the view frustrum of a real-time flyby. The *Multiscale Implicit TIN* utilises a similar strategy based on quadtrees for spatially indexing points, lines, polygons, complex polygons and composite objects constructed from these spatial objects. The *Multiscale Implicit TIN* not only dismisses the topology at each level of the TIN, but also between the levels of the Delaunay pyramid. Jones et al. (1994) provide an overview of the database schema for the *Multiscale Implicit TIN*

and demonstrate its use for variable scale access to polygonal search regions. The edges corresponding to surface features (such as ridges, valleys, and contours) are enforced as described in Section 3. De Floriani and Puppo (1988) give a detailed description of a *Constrained Delaunay Pyramid* construction algorithm that makes use of incremental point and edge insertion.

Several multiscale data storage schemes dedicated to linear data have been proposed (e.g. Ballard 1981, Becker et al. 1991, van Oosterom 1991). The approach adopted here is that of the Line Generalisation Tree (LG-tree). The LG-tree (Jones and Abraham 1987) is a multiresolution data structure in which vertex duplication is avoided, while providing selective access to those vertices required for a particular scale representation. Each level in the tree corresponds to a particular level of scale significance or contribution to the shape of the line as determined from the Douglas-Peucker line generalisation (Douglas and Peucker 1973). A hierarchy is then constructed in which, at the top level, all vertices required to represent the line in its most simplified form are stored. At the next level are stored intermediate vertices which when added to those at the higher level would represent the line to a pre-specified lateral error tolerance. Subsequent lower levels provide further degrees of detail. In the case of linear features that are closed or polygonal, then the feature is split in two at its extremes. The order of points within a linear feature can be maintained by either associating with each point a left and right control value which records the number of adjacent intermediate points at the next lower level or by storing a sequence number for each point which records its position in the original line. Although these methods introduce additional data in the form of indexing the vertices, it significantly reduces the data overheads of multiple line storage (Abraham 1988). An example of a line generalisation tree using sequence numbers is shown in Figure 9.

5 Digital Surface Modelling

Digital surface modelling is the term given to the incorporation of topographic features into the digital terrain model. It can be argued that the application of many digital terrain models to everyday GIS operations is flawed if the surface features of the terrain are ignored. For example, too often GIS users ignore the effect of features such as buildings and vegetation on intervisibility and viewshed calculations. This has been due to a combination of a shortage of accurate, large-scale, three-dimensional topographic data and the inability of traditional DTM data structures to integrate these topographic features. The former problem is being resolved with developments in spatial data acquisition, such as high-resolution aerial photography, satellite imagery, photogrammetry, and LiDAR laser scanning. Topographic features can be classified into two categories: man-made objects (buildings, roads, etc.) and natural phenomena (trees, hedgerows, etc.). Of the two, man-made objects are by far the easier to generalise and represent digitally, since their location, dimension and form remain relatively stationary and can be measured with a high degree of accuracy. Unlike man-made objects, vegetation is an ever-changing phenomenon, the boundaries of which are often difficult to define.

Large-scale 3D data are now more widely available at reasonable and affordable prices. However, the advantages of digital surface modelling will only be realised if digital terrain models can move from their constrained 2.5D approach to accommodate

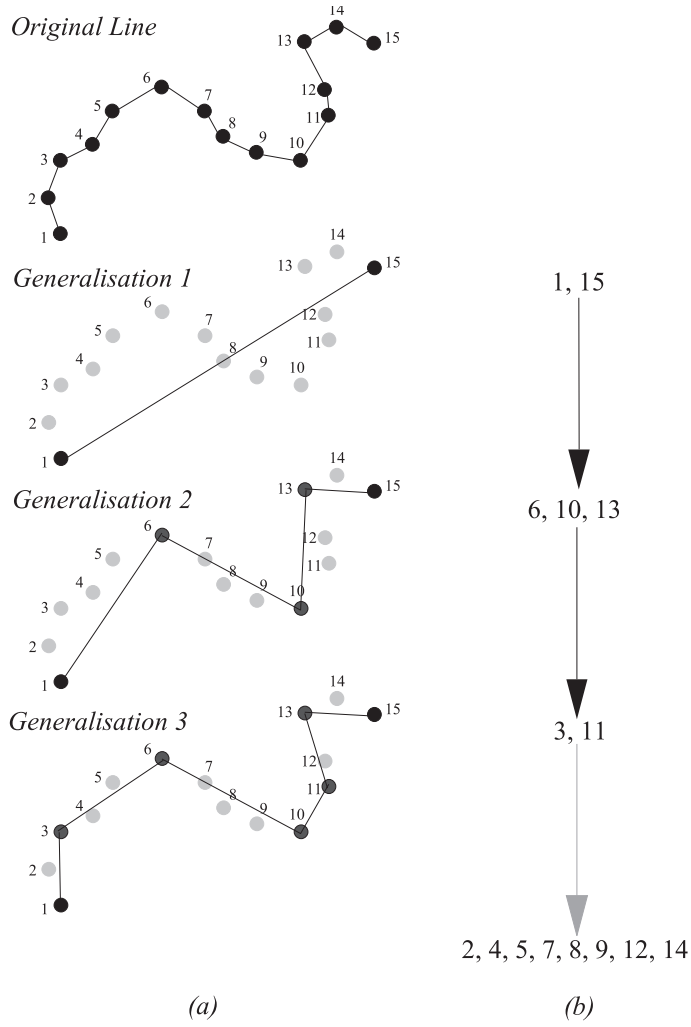


Figure 9 The Line Generalisation Tree. The example shown consists of three levels or generalisations and adopts the sequence number method (a) Original line and the results of applying the Douglas-Peucker algorithm with three different tolerance values (b) The LG-tree representation

these 3D data. The regular grid digital elevation model is flawed in this respect, as the resolution of the grid has to be set to the coordinate precision of the 3D data. For example, the UK's Environment Agency has specified two metres as the fixed lateral resolution for all LiDAR-derived digital surface models (DSMs) for their operations. As the original data are irregularly sampled, the data are interpolated and thus generalised or smoothed to fit this grid resolution. For some surface features a higher resolution would be beneficial, but for the underlying terrain elevations there is significant data redundancy.

An often-cited advantage of the TIN data model over the DEM is its ability to model sudden changes in the terrain by the inclusion of non-crossing breaklines and

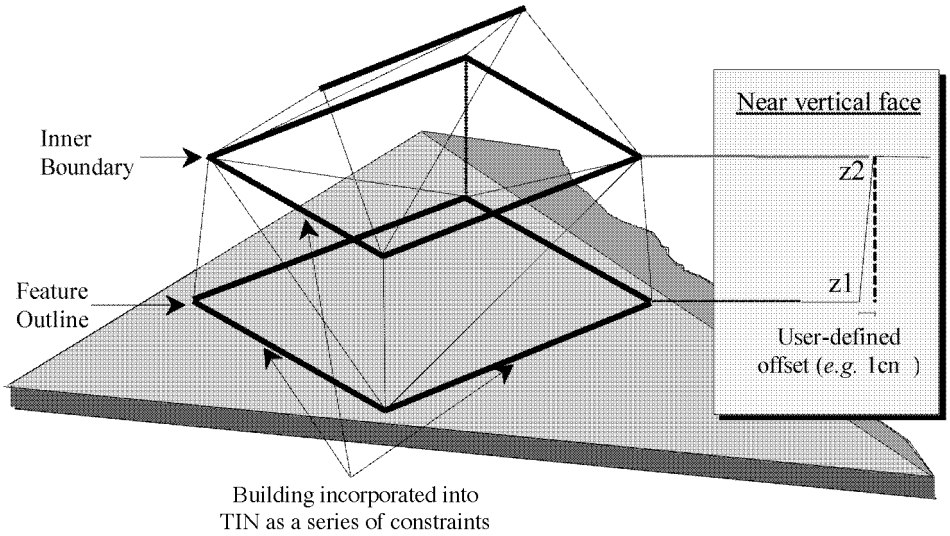


Figure 10 (a) Third and (b) Fourth level Multiscale Implicit TINs derived for a user-defined query window (view fustrum)

exclusion boundaries, thus forming a constrained triangulation. This is commonly put forward as a means of accurately representing abrupt natural changes in geomorphological phenomena, such as cliffs and ridge lines. The Earth's surface is rarely bare, rather it is populated by features that might also be thought of as representing abrupt changes in surface complexity. These include such objects as houses, trees and hedgerows for example. In the same way as sudden changes in geomorphological phenomena can be modelled by the use of breaklines, so too can topographic features.

The use of TINs as digital surface models is not widely documented within the GIS domain. To a large extent, digital surface models have been caught in *no man's land* between the true 3D systems employed in the geosciences and the CAD systems of the planners and landscape architects. In essence, the TIN can provide GIS users with a means of realising three dimensional modelling, rather than the 2D or $2\frac{1}{2}$ D limitations of the regular grid DEM. Kraak and Gazdzicki (1991) demonstrate the potential that the TIN offers as a *double elevation* model in the field of *Cartographic Terrain Modelling*. The biggest hurdle to overcome is the ability of the terrain model to cater for multi-valued points and surfaces, such as at the top and bottom of a vertical wall or cliff.

A topographic feature can be incorporated into the *Implicit TIN* by the insertion of two sets of edge constraints into the data model's *Edge File*. This in effect forces the feature outlines to be regarded in a similar manner to contour lines and ultimately, as constraints when the triangulation takes place. The first constraint is the feature perimeter or footprint, while the second is created by marginally shrinking the perimeter so that an inner boundary is created inside the original to simulate a vertical face or wall (Figure 10).

If having inserted a feature outline an intersection takes place between itself and an existing contour line, then the contour line must be split to avoid breaklines becoming

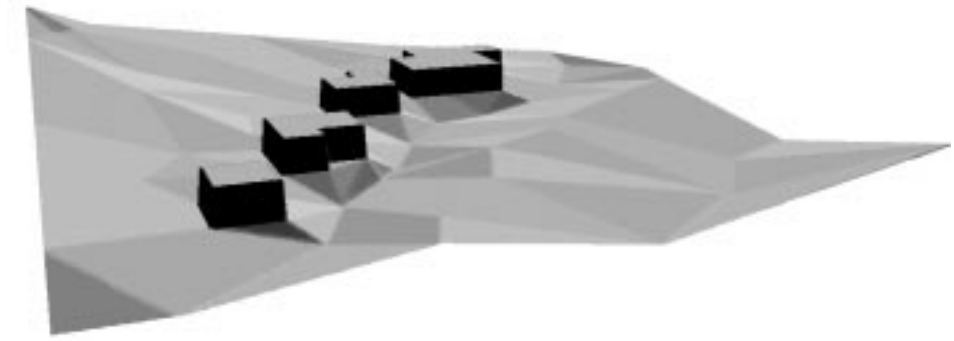


Figure 11 Cluster of buildings inserted into a TIN

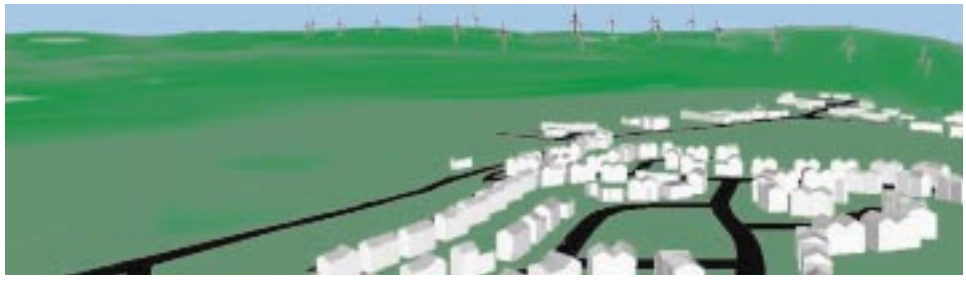


Figure 12 TIN digital surface model representation illustrating buildings with pitched roofs (Implicit TIN is visualised as a VRML scene)

crossed. Once given a suitable elevation, these constraints force the final triangulation to create triangles that are near vertical, depending on the precision which the user finds acceptable. This allows objects such as buildings and hedgerows to be represented accurately (Figure 11).

The methodology can be extended further to model more complex structures, i.e. the roofs of buildings. In reality, the variety of roof structures that exist means that the precise modelling of each building is often impractical. However a generic approach that produces more realistic roof-scapes than those of simple flat roofs, is to use the principal axis or to generate the skeleton (or medial axis) of each building polygon. These are inserted into the *Implicit TIN Edge File* as constraints and assigned heights equal to their respective feature. The heights of the internal wall constraints are then reduced accordingly in order to create the roof pitch (e.g. Figure 12).

Incorporating topographic features directly into the TIN requires that, after a feature polygon has been inserted into the *Edge File*, it is given an appropriate elevation. When modelling vegetation features, the heights of each of the footprint vertices are interpolated from the terrain elevations, while the interior vertices are also interpolated, but are increased by the height of the feature above ground level (Figure 13a). Buildings are represented by initially calculating the elevation of all of the feature's external vertices. Depending on the interpolation strategy required, the minimum, maximum or average elevation of all of these vertices is then calculated. The foundation of the building, represented by the external feature vertices, is then

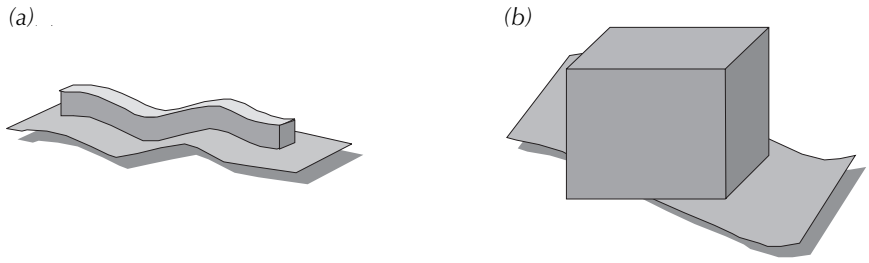


Figure 13 Placement of topographic features in the TIN (a) vegetation follows undulations of the terrain and (b) a building with flat foundations irrespective of the underlying terrain

assigned this single value. The building's structure is formed by assigning to all internal vertices this same elevation, plus the building's height above ground level (Figure 13b). There are instances where the roofs of terraced buildings follow the landscape in much the same way as vegetation. This can easily be modelled by identifying such buildings and adapting their attribute table to adopt vegetation modelling characteristics.

6 Multiscale Queries

The future of the TIN as a tool for digital terrain and surface modelling is dependent upon developers being able to evolve strategies for managing the high resolution data that are currently becoming more widespread. Crucially, the next generation of TINs will contain millions of vertices, rather than the thousands that today's terrain modellers are used to. As suggested in Section 4, we believe the key to solving this problem is to efficiently retrieve data into main memory, thus minimising the geographical domain of the TIN to the immediate problem in hand. The *Multiscale Implicit TIN* further extends this flexibility by offering users the opportunity to integrate or ignore the surface features on the terrain landscape.

In order to demonstrate the flexibility of the *Multiscale Implicit TIN*, we will consider some typical terrain modelling examples. In the first instance, the geographic area under consideration (Figure 14) is a 2×2 km area of eastern Cardiff, comprising housing estates, a retail complex, some major roads, the main line railway, areas of woodland and other vegetation, and a river. *Multiscale Implicit TINs* were derived from a variety of different data sources, including Ordnance Survey 1:50,000 and 1:10,000 scale contour DTMs, the Environment Agency's raw LiDAR data and Ordnance Survey large scale digital cadastral (LandLine) data. As an example, Figures 15 to 18 illustrate four explicit TIN representations at four different resolutions (8023, 19464, 50289, and 194717 vertices respectively) of increasing precision. As a measure of complexity, the TIN of Figure 18 consists of 388,750 triangles, while the full resolution model consists of well over a million vertices. Figure 18 clearly illustrates the linear constraints of the road network and the buildings. In essence, the TIN is finely tuned to the variability of the terrain and its surface features, whereas its equivalent regular grid DEM would possibly need to be at a metre resolution or less in order to capture the surface with similar precision.

Figure 14 The 2×2 km geographical area of Cardiff used for the subsequent Multiscale Implicit TINs of Figures 15 to 18

For the most part, any geographical query of a surface or a TIN will be concerned with only a limited subset of the data at any one time, such as a particular triangle, an edge, or a range of triangles. More general queries, such as the calculation of a viewshed or the derivation of a contour map are problems that can be partitioned into manageable segments and their solutions combined. This is essentially how most GIS handle intensive processing tasks. However, with the *Implicit TIN* the data and topology are retrieved into main memory only when required. The full TIN topology is impossible to maintain in main memory, even with today's advances in processors and RAM. As such, the GIS of the future will need to consider more efficient strategies for disk-based retrieval of data for partitioning solutions. For example, Figures 19 and 20 illustrate the retrieval of TIN vertices and the generation of the surface topology for a user query based on a view fustrum, possibly for landscape visualisation or part of a segmented viewshed calculation. These four TIN queries correspond to the four

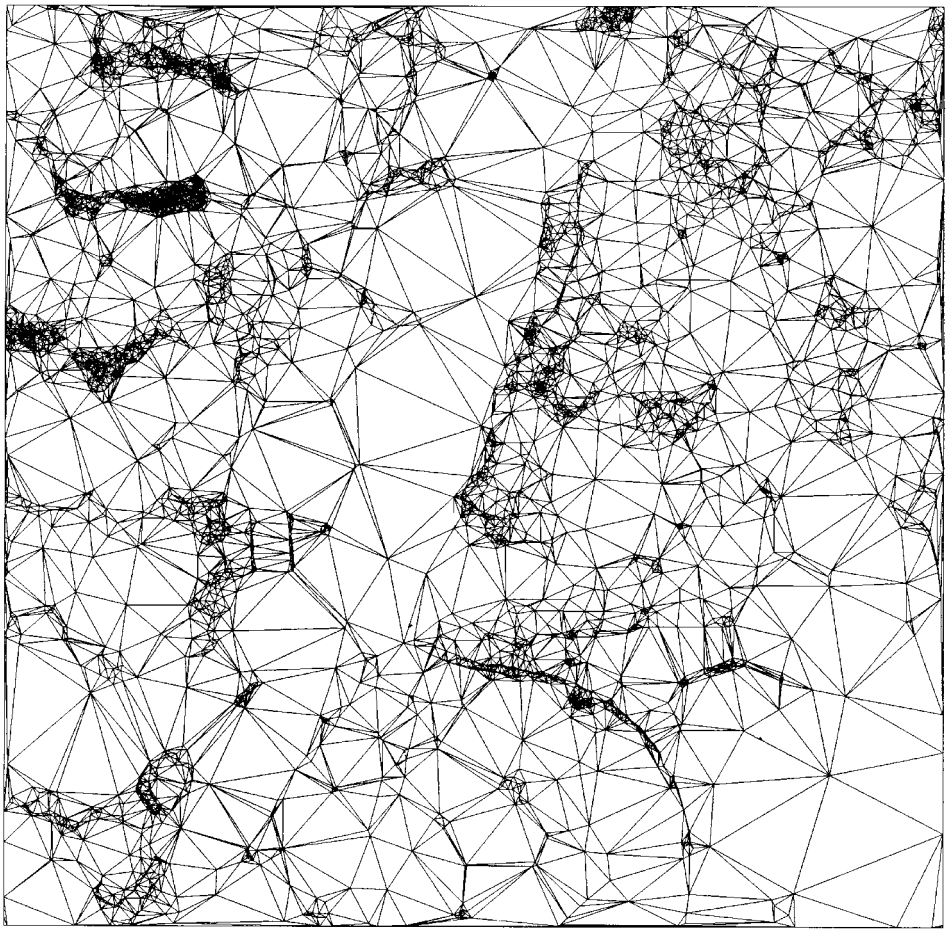


Figure 15 First level Multiscale Implicit TIN of Figures 14 defined from 8,023 vertices

multiscale TINs of Figures 15 to 18. In each case, the user query window is defined, which is then intersected with the quadtree index of the *Implicit TIN* for retrieving the relevant data. A more localised indexing scheme is then used for the retrieved data for minimising the search time during triangulation and any subsequent operations. By minimising the data held in main memory, everyday TIN operations are faster than their full-TIN equivalent.

This philosophy can be taken further for queries such as the retrieval of a surface cross-section. In this instance, the only data actually required are the vertices that form the triangles that are intersected by the cross-section. Any other data are superfluous and may hinder the time taken to interpolate the cross-section. Kidner and Jones (1991) demonstrated an algorithm for retrieving cross-sections within a user-defined geographical region, but the *Implicit TIN* is re-constructed for all the vertices that are spatially indexed within this region. Smith et al. (1992) refined this algorithm to retrieve only the minimal set of triangles needed to derive the terrain cross-section. This is illustrated in Figure 21 for a cross-section derived through the four TINs of Figures

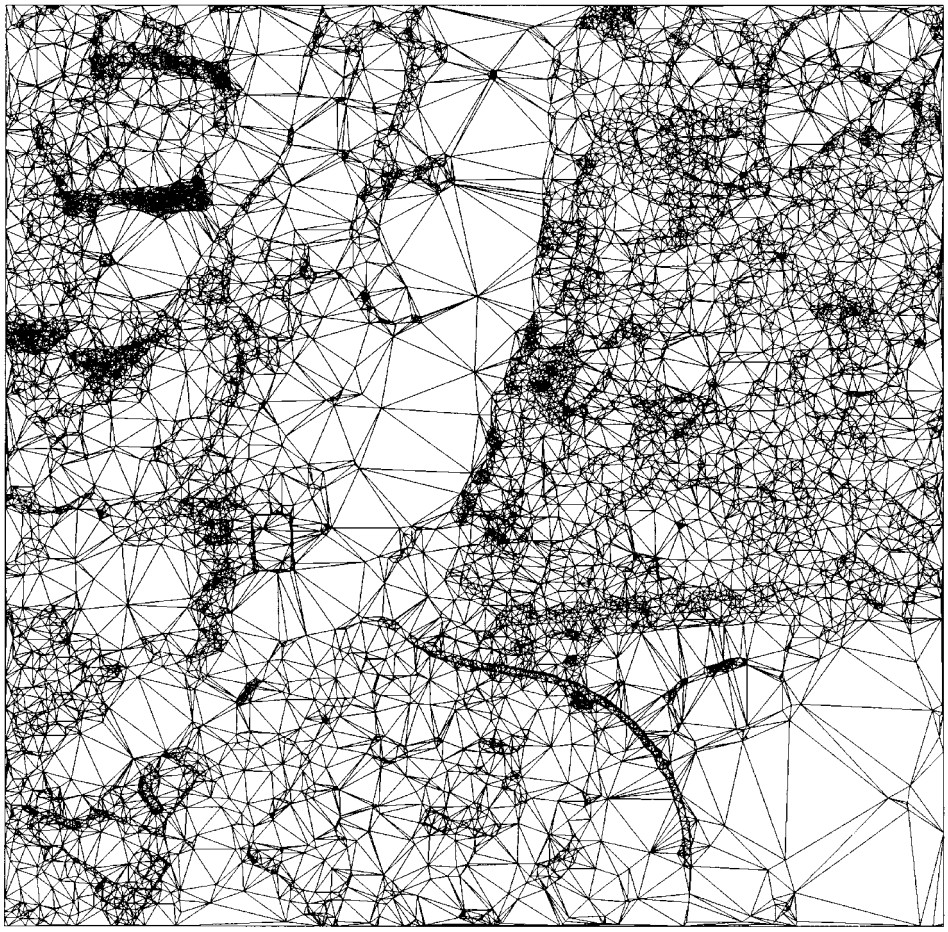


Figure 16 Second level Multiscale Implicit TIN of Figure 14 defined from 19,464 vertices

15 to 18, while Figure 22 illustrates the terrain profiles derived from these different multiscale TINs.

Further arguments to support the principle of the *Multiscale Implicit TIN's* approach to terrain modelling have been raised by the proponents for surface visualisation within the computer graphics domain. For example, de Berg and Dobrindt (1995) argue that it is insufficient for rendering purposes to use only one level of detail at a time, but data from the different levels should be merged into a single representation commensurate with a particular view (or operation). This argument suggests that the levels cannot be completely independent and has given rise to arguments supporting the nested *Hierarchical TIN* (Abdelguerfi et al. 1998). However, multiscale *Delaunay Pyramids* can maintain the necessary topology to make access at different levels quite efficient. For the *Multiscale Implicit TIN*, the implied relationship between vertices at different levels is maintained by specifying the scale or error tolerance of data. As such, the retrieval of a TIN surface, even at one fixed scale will take account of the data represented at lower levels, so that the

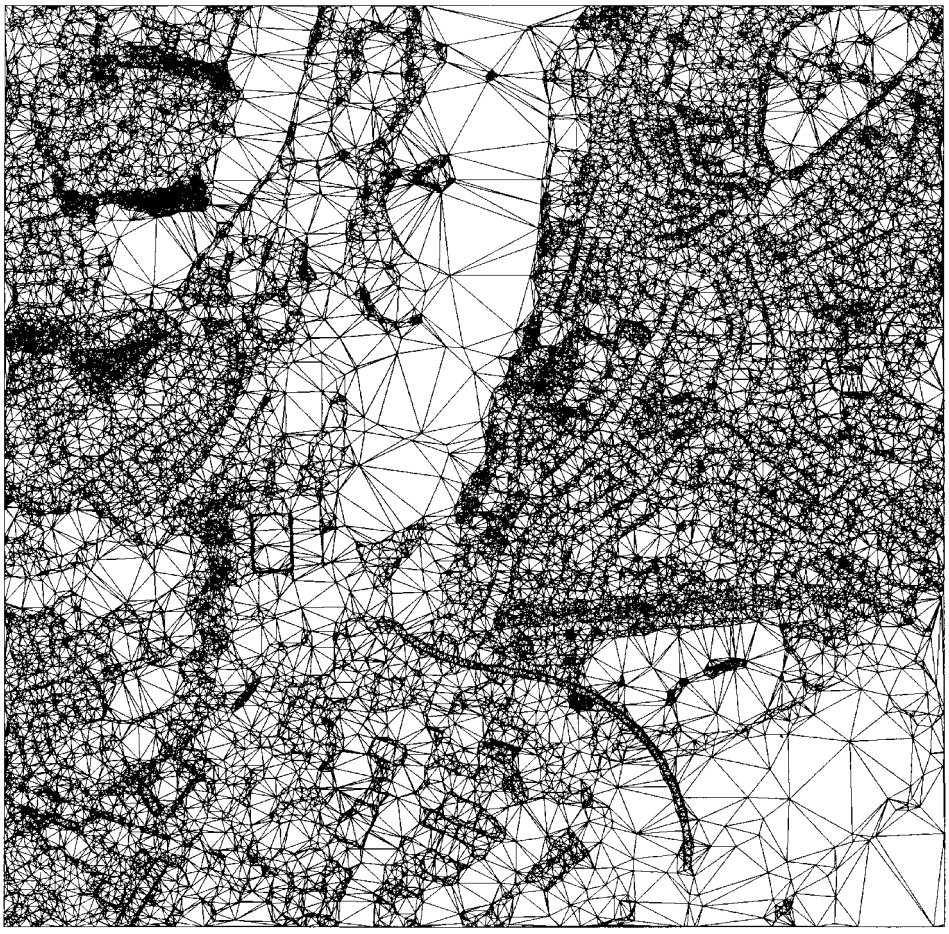


Figure 17 Third level Multiscale Implicit TIN of Figure 14 defined from 50,289 vertices

integration of data at multiple levels of detail is easily supported. Perhaps the biggest issue is ensuring that the query window can now also be defined at multiple levels or with different ranges. For example, Figure 23a illustrates the concept of the integrated levels-of-detail for a TIN derived at different vertical error tolerances, while Figure 23b emphasises the lower detail and hence lower rendering time required at more distant locations. However, although such visualisations are acceptable for terrain visualisation or flight simulation, there is a wider implication for maintaining the integrity of surface objects which cut across the query windows of different levels (Figure 24). Currently, the *Multiscale Implicit TIN* just triangulates between the data retrieved at each level, thus providing a satisfactory, if not elegant solution. While this is fine for the terrain data itself, the automatic generalisation and maintenance of multiple representations of topographic features is less well understood. Jones et al. (1994) give a more detailed overview of the spatial indexing for querying the *Multiscale Implicit TIN*.

Figure 18 Fourth level Multiscale Implicit TIN of Figure 14 defined from 194,717 vertices

7 Implicit TIN Performance

The *Multiscale Implicit TIN* provides a unique approach to digital terrain and surface modelling, which means that direct comparisons with single scale and hierarchical TINs are unsatisfactory and meaningless. The fact that most of the topology of the *Implicit TIN* has to be derived at run-time may not appeal to those users who will only need to work at a single, fixed scale. Similarly, the *Implicit TIN* cannot compete with a traditional hierachical TIN if real-time visualisations are required, such as in flight simulation. However, there are many other scenarios where the *Multiscale Implicit TIN* will out perform its explicit counterparts. A hydrologist may require a TIN bereft of its topographic features, while a radiocommunications planner may require a full topographic TIN. There is no need to maintain multiple explicit representations with the *Multiscale Implicit TIN*. Not only does this allow significant storage savings, but it

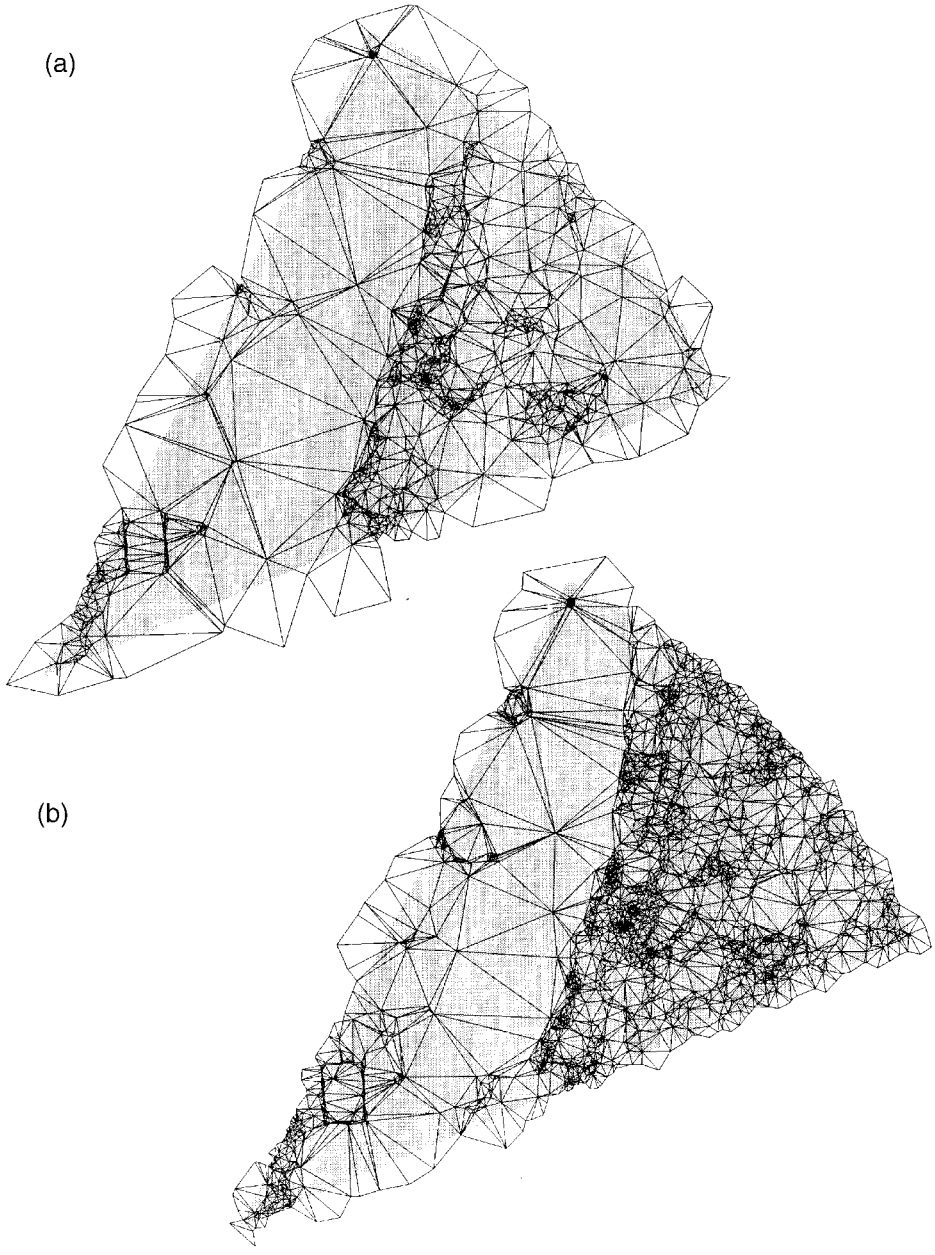


Figure 19 (a) First and (b) Second level Multiscale Implicit TINs derived for a user-defined query window (view fustrum)

provides flexibility in integrating selected topographic features with a terrain model at user specified levels of detail. Thus the constraints introduced by the selected topographical features are not predetermined, as they would be in a stored constrained explicit TIN.

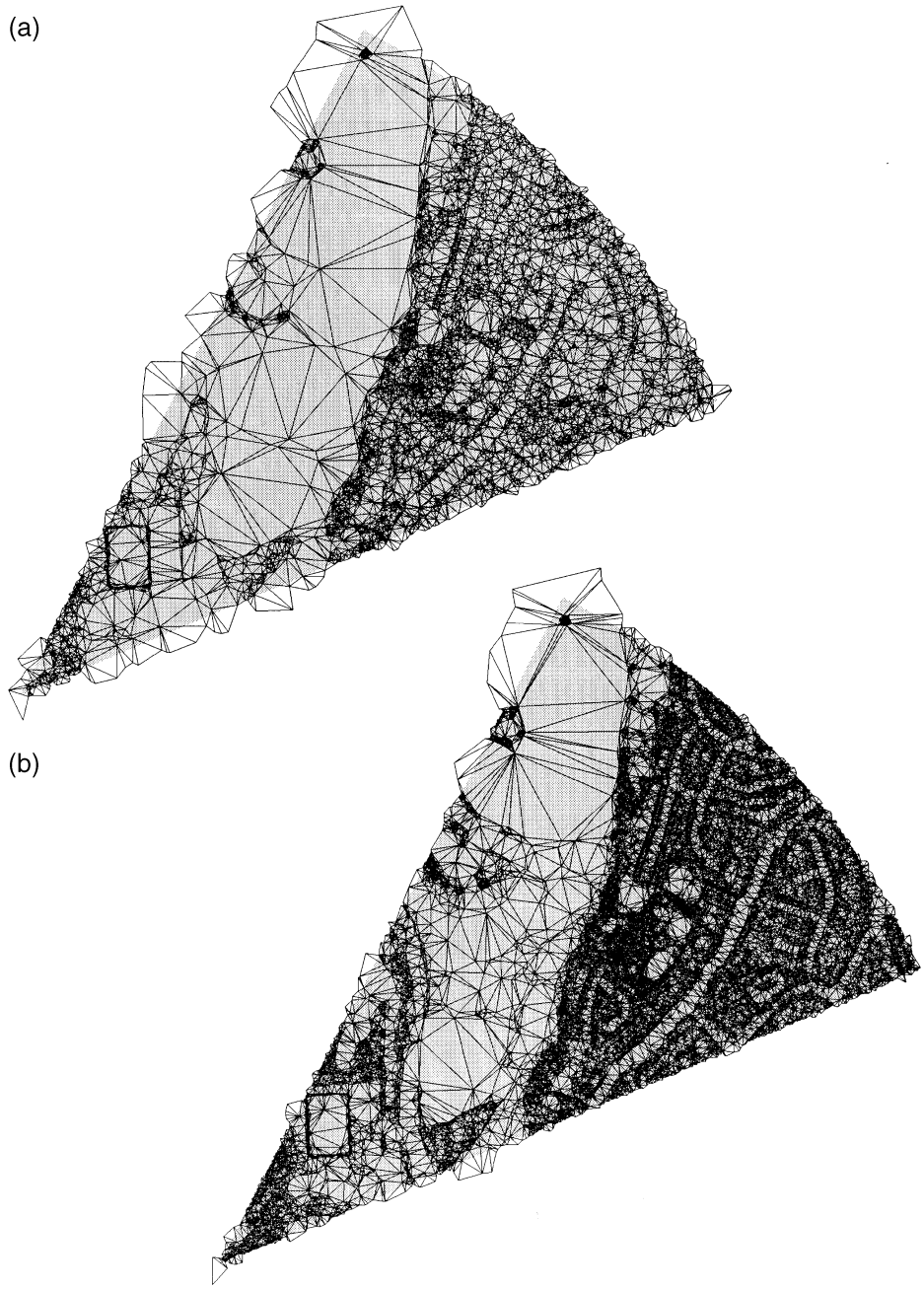


Figure 10 (a) Third and (b) Fourth level Multiscale Implicit TINs derived for a user-defined query window (view fustrum)

The usefulness of the *Multiscale Implicit TIN* will depend, for many applications, on the ability to reconstruct the correct constrained Delaunay triangulation for a given query region within a satisfactory time, the length of which will relate to the specific needs of the particular application. The major time penalty introduced by the

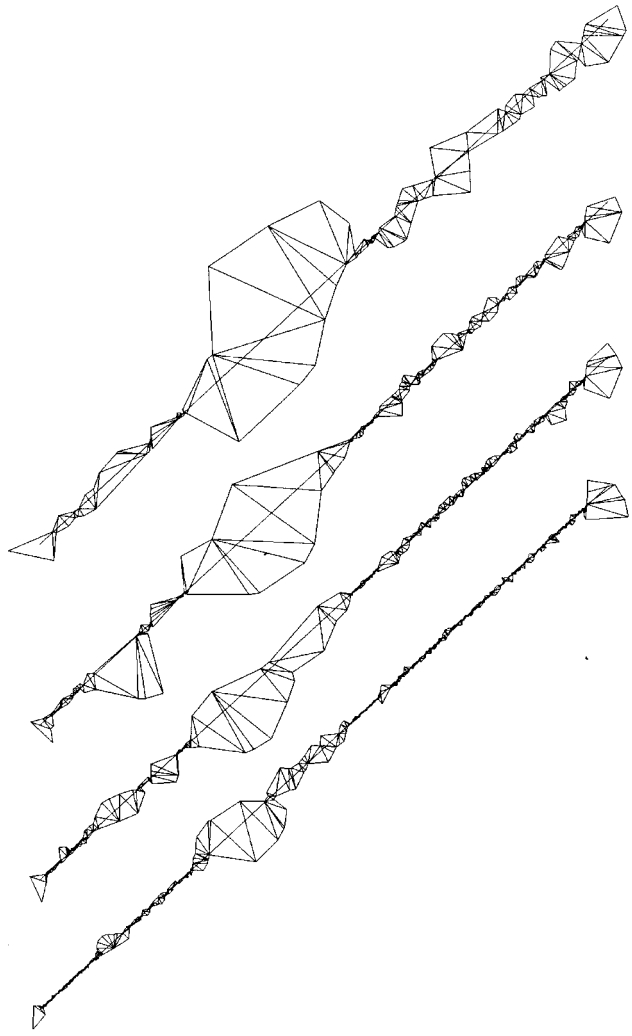


Figure 21 Cross-sectional profile interpolation through the four Multiscale Implicit TINs of Figures 15 to 18

Multiscale Implicit TIN is that of having to reconstruct the constrained Delaunay triangulation. The algorithm used in this system has a worst case time complexity of $O(n \log n)$, where n is the number of points to be triangulated. This represents an upper bound on time for any serial Delaunay triangulation algorithm (constrained or unconstrained), although some parallel algorithms improve on this, with $O(\log n)$ reported by El Gindy (1990). For some applications, such as terrain profile reconstruction (Figures 23 and 24) for intervisibilrty analysis, the amount of data to be retrieved and triangulated is negligible when compared with the time taken to read an explicit TIN into main memory. For today's datasets, the feasibility of maintaining explicit TINs of millions of vertices in main memory is severely hindered. The accuracy of the *Multiscale Implicit TIN* is not considered here, but is addressed

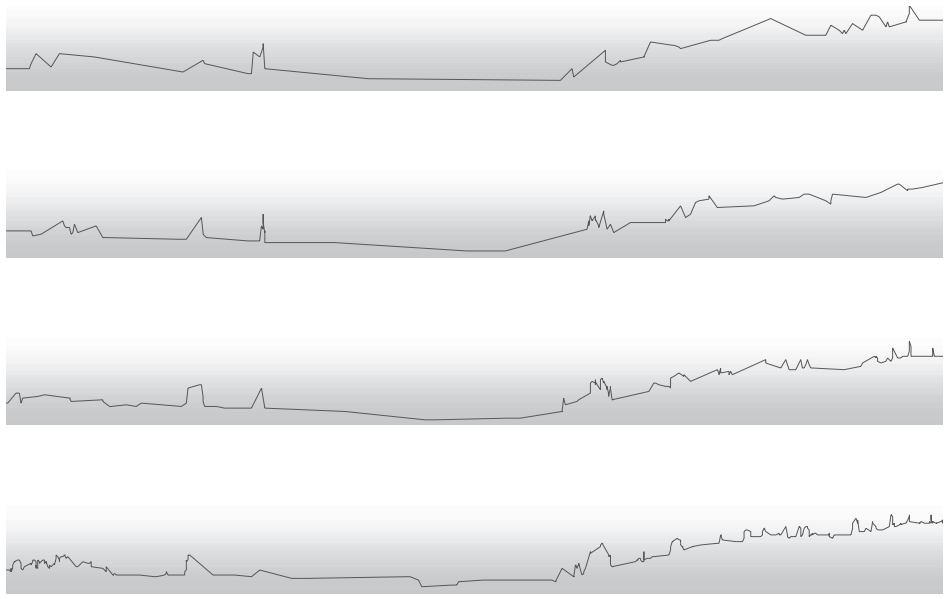


Figure 22 Cross-sectional profile re-constructions through the four Multiscale Implicit TINs of Figures 15 to 18

in the accompanying paper by Kidner et al. (2000) for the application of visibility analysis.

8 Conclusions

The paper has presented an efficient triangulated data storage scheme that has considerable potential for representing topographic surfaces in a multiscale database for a range of terrain modelling functions. Multiscale databases for GIS raise many challenging issues relating to the integration of data of different quality from different spatial models and to the automated generalisation of the retrieved data. This will no doubt be a major topic of GIS research over the next few years as a range of more affordable 3D topographic data and larger scale DTMs become available. The founding principle of the *Multiscale Implicit TIN* is that the data structure should be derived at run-time by the requirements of the user and the specific queries asked of the system. The range of today's digital terrain modelling applications is a good example of how GIS can be used for global or local, exploratory or precise analysis. A single scale data model cannot meet all of these demands, while the main memory and processing requirements of large scale digital surface models derived from products such as laser altimetry are beyond the means of even today's high specification workstations. The key to solving this problem is to efficiently retrieve only the pertinent data into main memory, thus minimising the geographical domain and excluding redundant entities. This can be achieved with good spatial indexing and adapting the search to the specific user query. The *Multiscale Implicit TIN* rises to this challenge by providing a flexible framework for digital surface modelling that allows multiscale terrain models to be integrated with 3D topographic features. This has been

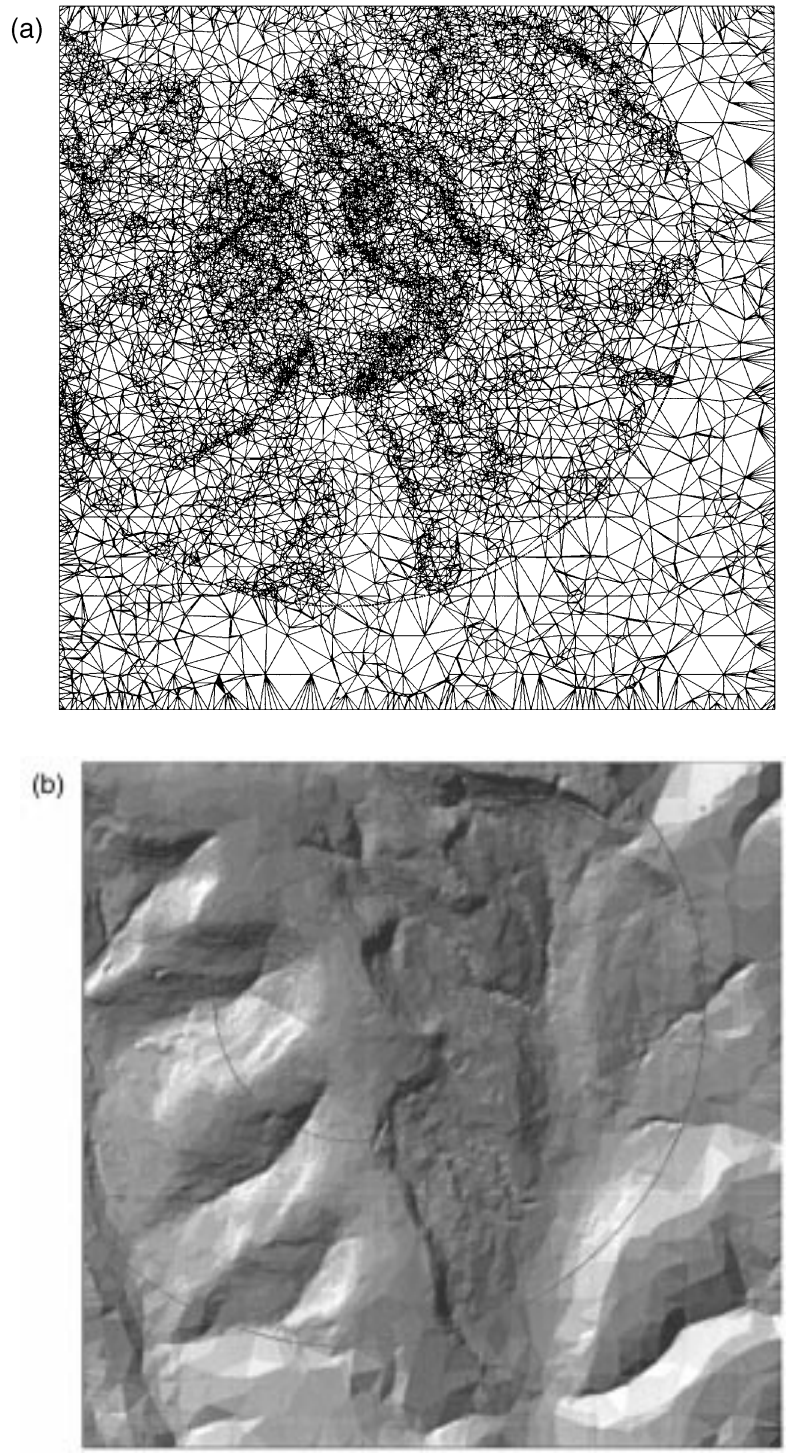


Figure 23 (a) Multiple level-of-detail TIN and (b) corresponding hill-shaded representation for 1 km and 2.5 km radii query windows

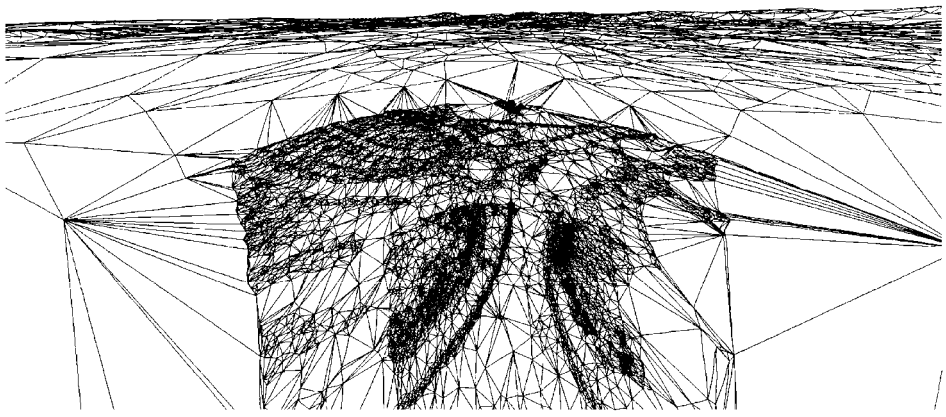


Figure 24 Integrated level-of-detail TIN derived from the multiple data sources of a Multiscale Implicit TIN centred at the roundabout of Figure 14 and looking westward. The field-of-view is exaggerated to illustrate the different data sets

illustrated for some general multiscale queries, while Kidner et al. (2001) further analyse and quantify the accuracy of the *Multiscale Implicit TIN* for intervisibility analysis.

References

Abdelguerfi M, Wynne C, Cooper E, and Roy L 1998 Representation of 3-D elevation in terrain databases using hierarchical triangulated irregular networks: A comparative analysis. *International Journal of Geographical Information Science* 12: 853–73

Abraham I M 1988 Automated Cartographic Line Generalisation and Scale Independent Databases. Unpublished PhD Dissertation, Department of Mathematics and Computing, The Polytechnic of Wales

Ballard D H 1981 Strip trees: A hierarchical representation for curves. *Communications of the ACM* 24: 310–21

Becker B, Six H-W, and Widmayer P 1991 Spatial priority search: An access technique for scaleless maps. *ACM SIGMOD Record* 20: 128–37

Bertolotto M, De Floriani L, and Marzano P 1994 An efficient representation for pyramidal terrain models. In Pissinou N and Makki K (eds) *Proceedings of the Second ACM Workshop on Advances in Geographic Information Systems*. Gaithersburg, MD: 152–9

Clarke A L, Gruen A, and Loon J C 1982 The application of contour data for generating high fidelity grid digital elevation models. In *Proceedings of the Fifth International Symposium on Computer-Assisted Cartography (Auto-Carto 5)*: 213–22

de Berg M and Dobrindt K T G 1995 On Levels of Detail in Terrains. WWW document, ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-1995/1995-12.ps.gz

De Floriani L 1987 Surface representations based on triangular grids. *The Visual Computer* 3: 27–50

De Floriani L and Puppo E 1988 Constrained Delaunay triangulation for multiresolution surface description. In *Proceedings of the Ninth International Conference on Pattern Recognition*, Italy, November, 1988: 566–9

De Floriani L 1989 A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications* 9: 67–78

De Floriani L 1995 Multiresolution modeling in geographical information systems. Unpublished paper Presented at GISRUK '95, University of Newcastle: Newcastle-upon-Tyne, UK

Douglas D H and Peucker T K 1973 Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *The Canadian Cartographer* 10: 112–22

El Gindy H 1990 Optimal parallel algorithms for updating planar triangulations. In *Proceedings of the Fourth International Symposium on Spatial Data Handling*. Columbus, OH, International Geographical Union: 200–8

Heckbert P S and Garland M 1997 Survey of Polygonal Surface Simplification Algorithms. WWW document, ftp://ftp.cs.cmu.edu/afs/cs/project/anim/ph/paper/multi97/release/heckbert/simp.pdf

Heller M 1990 Triangulation algorithms for adaptive terrain modelling. In *Proceedings of the Fourth International Symposium on Spatial Data Handling*. Columbus, OH, International Geographical Union: 163–74

Hoppe H 1996 Progressive meshes. In *SIGGRAPH '96 Computer Graphics Proceedings*: 99–108

Hoppe H 1997 View dependent refinement of progressive meshes. In *SIGGRAPH '97 Computer Graphics Proceedings*: 189–98

Hoppe H 1998 Smooth view-dependent level-of-detail control and its application to terrain rendering. In Ebert D, Hogan H, and Rushmeier (eds) *Smooth View-dependent Level-of-Octail Control and Its Application to Terrain Rendering*. Washington DC, IEEE: 35–42

Jones C B and Abraham I M 1987 Line generalisation in a global cartographic database. *Cartographica* 24: 32–45

Jones C B, Kidner D B, and Ware J M 1994 The implicit triangulated irregular network and multiscale spatial databases. *The Computer Journal* 37: 43–57

Kidner D B 1991 Digital Terrain Models for Radio Path Loss Calculations. Unpublished PhD Dissertation, Department of Mathematics and Computing, The Polytechnic of Wales

Kidner D B and Jones C B 1991 Implicit triangulations for large terrain databases. In *Proceedings of the Second European Conference on GIS (EGIS '91)*, Brussels, April, 1991: 534–46

Kidner D B, Sparkes A J, Dorey M I, Ware J M, and Jones C B 2001 Visibility analysis with the Implicit TIN. *Transactions in GIS* 5: In Press

Kofler M, Gervautz M, and Gruber M 1998 The Styria flyover: LOD management for huge textured terrain models. In *Proceedings of Computer Graphics International 98,* Hanover: 11p

Kraak M J and Gazdzicki J 1991 Triangulation based modelling of spatial objects in relation to the terrain surface. In *Proceedings of the Second European Conference on GIS (EGIS '91)*, Brussels, April, 1991: 564–72

Kumler M 1994 An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). *Cartographica* 31: 1–99

Lee J 1991a Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *International Journal of Geographical Information Systems* 5: 267–85

Lee J 1991b Analyses of visibility sites on topographic surfaces. *International Journal of Geographical Information Systems* 5: 413–29

Lindstrom P, Koller D, Ribarsky W, Hodges L F, Faust N, and Turner G 1996 Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96 Computer Graphics Proceedings*: 109–18

McCullagh M J and Ross C G 1980 Delaunay triangulation of a random data set for isarithmic mapping. *The Cartographic Journal* 17: 93–9

Mitas L and Mitasova H 1999 Spatial interpolation. In Longley P A, Goodchild M F, Maguire D J, and Rhind D W (eds) *Geographical Information Systems: Principles, Technical Issues, Management Issues, and Applications*. New York, Wiley: 481–92

Moore K, Dykes J, and Wood J 1999 Using Java to interact with geo-referenced VRML within a virtual field course. *Computers and Geosciences* 25: 1125–36

Nelson R C and Samet H 1986 A consistent hierarchical representation for vector data. *SIGGRAPH* 20: 197–206

Peucker T K, Fowler R J, Little J J, and Mark D M 1978 The triangulated irregular network. In *Proceedings of the Digital Terrain Models (DTM) Symposium*. Baltimore, American Congress of Surveying and Mapping: 516–40

Reddy M, Lecelerc Y, Iverson L, and Bletter N 1999 TerraVision II: Visualizing massive terrain

databases in VRML. *IEEE Computer Graphics and Applications* 19: 30–8

Rippa S 1992 Long and thin triangles can be good for linear interpolation. *SIAM Journal of Numerical Analysis* 29: 257–70

Smith D H, Jones C B, and Kidner, D B 1992 *Terrain and Clutter Databases for Radio System Planning*. Pontypridd, University of Glamorgan Commercial Services Radiocommunications Agency Technical Report

Sparkes A J 2000 Quantifying the correctness of GIS line-of-sight predictions on the triangulated irregular network. Unpublished PhD Dissertation, School of Computing, University of Glamorgan

van Oosterom P 1991 The reactive-tree: a storage structure for a seamless, scaleless geographic database. In *Proceedings of the Tenth International Symposium on Computer-Assisted Cartography (Auto-Carto 10)*. Baltimore, American Congress of Surveying and Mapping: 393–407

Ware J M 1998 A procedure for automatically correcting invalid flat triangles occurring in triangulated contour data. *Computers and Geosciences* 24: 141–150

Weibel R 1997 Digital terrain modelling for environmental applications: A review of techniques and future trends. In *Proceedings of the Third Joint European Conference and Exhibition on Geographical Information*, Vienna, 16–18 April. Amsterdam: IOS Press: 464–74

Woo T C 1985 A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications* 5: 19–27