Applications of Computational Geometry to Geographic Information Systems

Leila De Floriani[1], Paola Magillo[1], Enrico Puppo[1]

[1] Dipartimento di Informatica e Scienze dell'Informazione - Università di Genova
Via Dodecaneso, 35 - 16146 Genova, Italy
Email: {deflo,magillo,puppo}@disi.unige.it

Contents

1 Introduction

During the last decade, Geographical Information Systems (GISs) have gained a large impact over a variety of important application fields [171]. According to Goodchild, Kemp, and Poiker [119] "a GIS can be seen as a system of hardware, software and procedures designed to support the capture, management, analysis, modeling and display of spatially-referenced data for solving complex planning and management problems."

Geographic data are characterized by spatial properties (location, shape, size) and non-spatial properties, called attributes, usually expressed in textual or numerical form. GIS as a discipline involves many different issues, such as hardware and software equipment for data acquisition, data standards, storage and transmission, database management issues (such as integrity and consistency), etc. In this chapter, we consider only issues related to representation and processing of the *geometric aspects of geographic data*, with special emphasis on the application of computational geometry techniques.

In the past, classical information retrieval methods were adopted to handle geographic data, while the importance of geometric aspects was often underestimated: geometric problems in GIS were solved through *ad hoc* empirical methods. Such solutions often suffered from lack of foundations, both from a geometric point of view (e.g., the result of an operation defined just as the output of a certain algorithm), and from a computer science perspective (e.g., no complexity analysis), and sometimes had poor practical performances.

The needs for a solid theoretical background and for high performances in geometric reasoning has become urgent in GIS. This makes GIS a field of primary importance for application of computational geometry. Representation and manipulation of spatial entities in a geographic information system involve modeling and computational issues. The need to store and process huge data sets yields a demand for data structures and algorithms that achieve a good tradeoff between high computational efficiency and low storage space.

In this chapter, we review classical as well as new problems in geometric modeling and computational geometry arising in GIS. We follow a broad classification of classical geometric data in a GIS into *map* data and *terrain* data. Such classification is not standard in the GIS community, but it is convenient here to identify data characterized by different spatial dimensionality, and different classes of problems. Map data are located on the surface of the Earth and are basically two-dimensional, i.e., they are points, lines, and polygonal regions, which are combined together to form either subdivisions or arrangements, sometimes organized into layers. Terrain data are related with the three-dimensional configuration of the surface of the Earth. The geometry of a terrain is modeled as a $2\frac{1}{2}$-dimensional surface, i.e., a surfaces in 3D space described by a bivariate function. In addition to these two kinds of classical GIS data, we cover also new research issues in GIS related to the management of fully three-dimensional data.

The first part of this chapter is devoted to map data. The representation of map data involves designing models and data structures capable of encoding geometric structures more general than those studied in the geometric modeling literature, like regions with holes, non-regular structures, isolated points, and line features. Spatial indexes also play an important role in the design of efficient methods for geometric queries of map data. Geometric aspects of map data processing involve query operations (e.g., point location, range search); spatial joins (map overlay); and more specific operations based on geometric search, such as map generalization, map labeling, map conflation (i.e., reconciliation of two variants of a map of the same area), computation of optimal paths.

Terrain representation and processing are an even richer source for geometric problems. The construction of terrain models involves conversions among raw point and line data, raster grids, triangulated surfaces, and contour line representations. Algorithms for converting among different representations involve several problems in computational geometry, from point location to computation of Voronoi diagrams, Delaunay and other optimal triangulations of points and lines. Terrain analysis and visualization include several geometric operations, such as contour line extraction, computation of drainage networks and topographic features, extraction of models at resolution variable on the domain, computation of paths on a surface. Problems related to terrain visibility, which have been extensively studied in computational geometry, are of primary interest in geographic data processing.

The rest of this chapter is organized as follows. In Section 2 we deal with map data modeling in GIS. We first define geographic maps mathematically in terms of spatial entities and relations, and then introduce two major map models used in GIS (the raster and the vector model). Since vector map data are much a stronger source of applications for computational geometry techniques, the attention is focused on such kind of map representations. Computational issues related to map data processing and analysis are treated in Section 3.

In Section 4, we consider terrain data modeling and processing. We describe classical terrain models used in GIS and review basic issues in converting terrain data among the various possible formats. The survey also covers approximate and multiresolution terrain models. In Section 5, we examine some of the most classical terrain analysis problems: visibility problems, extraction of topographic features, computation of drainage networks and determination of optimal paths on a terrain. Issues on trhee-dimensional GIS are briefly discussed in Section 6. Finally, Section 7 gives a summary of the chapter, by providing a list of computational geometry techniques and tools that can be useful in GIS, as well as a list of open geometric problems in GIS.

This survey is far from being complete. Several problems and geometric structures are just mentioned here and the reference is given to available publications, or to other chapters of this book. In particular, we do not cover parallel algorithms (treated in Chapter 4) and raster representations, since techniques used to manipulate raster data are more typical of digital geometry [165] than of classical Computational Geometry.

2 Map Data Modeling

2.1 Two-Dimensional Spatial Entities and Relations

Two-dimensional entities considered in the context of geographic data processing are *points*, *lines*, and *regions* embedded in a two-dimensional space. Datasets covering small portions of the Earth are often projected onto a flat surface and modeled using classical Cartesian coordinates. Several kinds of projections can be used [119]: azimuthal projections using a plane tangent to the Earth in the center of the area to be mapped, conical projections, cylindrical projections using a cylinder tangent to the Earth along the Equator (Mercator projection) or along a line of longitude (Transverse Mercator projection).

A more challenging problem is defining a global reference system. Various conventions are used for such purpose. One approach involves subdividing the Earth surface into zones, each of which is projected

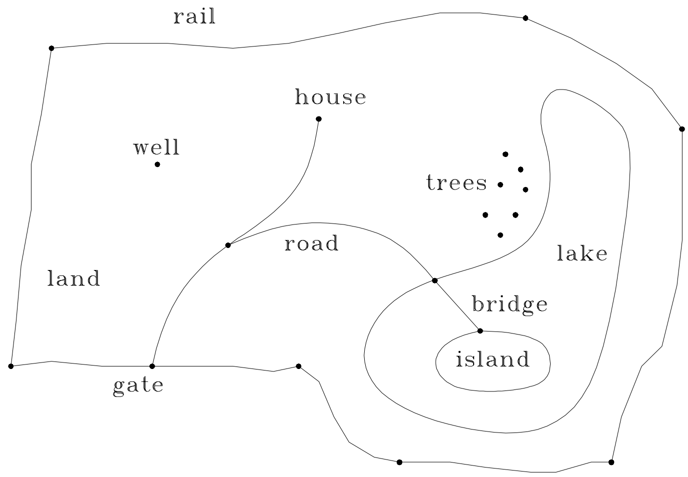

Figure 1: A toy example of a geographic map.

separately (the Universal Transverse Mercator uses 60 different zones); other approaches use a spherical coordinate system based on long-lat coordinates, or projection onto an ellipsoid [75, 76, 182]. However, in the sequel we will always assume the local convention that map data are embedded in a Euclidean plane.

GIS entities are more general than those usually adopted in Computational Geometry. In some cases, lines may not be simple; regions may not be simply connected (i.e., they may contain *holes*), and may have internal *features*, i.e., dangling lines (*cuts*), and isolated points (*punctures*). Figure 1 shows a toy example of a geographic map. The presence of holes and features may affect the design of data structures and algorithms for processing geographic entities. Most computational methods in GIS deal only with piecewise-linear geometric entities. Although data structures and techniques for representing and manipulating curved entities also exist, they are beyond the scope of this chapter.

Spatial relations are defined on pairs of spatial entities, and depend on their relative positions in space. There is a flourishing literature on the definition, classification, evaluation, and study of invariants of spatial relations [42, 43, 58, 85, 86, 130, 155, 156, 291, 292]. In spite of all such work, a homogeneous and complete characterization (e.g., based on an algebraic approach) is still missing. It is not uncommon to find practical contexts and applications for which none of the models proposed in the literature is satisfactory. However, spatial relations are usually classified as follows:

- *topological relations* are characterized in terms of the geometric intersections between boundaries, interiors, and complements of entities: examples are incidence, adjacency, containment, overlapping, etc. (see [42] for a survey);
- *metric relations* are defined in terms of distance between entities: examples are nearest neighbors and range queries;
- *directional relations* are characterized in terms of relative position of entities with respect to some oriented direction: examples are before, after, right-of, left-of, north, south, etc.;
- *order relations* are defined by properties that induce a partial order on sets of entities. An example is given by containment [157].

Spatial relations play a central role in GISs because many tasks involve the evaluation of spatial relations between a query object and a collection of spatial entities.

A broad classification of spatial datasets, which is convenient for both modeling and computational purposes, is as follows:

- *collections* of spatial entities (sometimes organized into distinct *layers*) that can take any possible relative position on the plane, and intersect arbitrarily;
- *subdivisions* of a compact planar domain into a set of non-overlapping entities.

Collections may define arrangements which are more general than those studied in computational geometry, since they can contain irregular entities of any shape and dimension. Similarly, also subdivisions are more general than those used in computational geometry, since they may contain multiply-connected regions with features. Subdivisions are simpler than collections. The combinatorial complexity of a subdivision is linear in the total size of its entities, while the arrangement induced by a collection may have a quadratic complexity. A pair of entities in a collection may assume any possible topological relationship, while possible topological relations in the context of a subdivision just consist of adjacency and incidence among its constituent entities, and inclusion of features.

2.2 Raster and Vector Models

There is a long-running debate in the GIS community on whether it is better to represent spatial information based on a vector or on a raster model. Neither model appears to be superior in all tasks. As outlined by Frank [106]: "A traditional view is to differentiate between an entity based view - space is constructed from objects that fill space - and a space oriented view, where each point in space has some properties. This view is philosophically well established....". The vector and raster approaches seem to correspond to these two alternative views of spatial concepts.

In a *vector* model, spatial entities are represented explicitly through their geometry and attributes. Either collections or subdivisions may be represented, and topological relations among entities can be stored explicitly. Explicit representation can support an arbitrarily good level of accuracy. The vector model is obviously entity-based, hence it is well suited to access a spatial database by using spatial entities as search keys. On the other hand, efficient space-oriented access may require sophisticated search structures and techniques.

Early representations in vector data modeling, called *spaghetti models*, are collections of unrelated simple polygonal lines and points. In a spaghetti model, regions are not explicitly encoded, while they are implicitly described by their boundaries. A spaghetti model may encode either a collection or a subdivision, since it is just a collection of entities without any stored relation. The drawback is an almost total lack of support for efficient query processing. More effective data structures are discussed separately in the following subsections.

In a *raster* model, the domain is regularly subdivided into a large number of atomic regions, similar to the pixels in a digital image. Each pixel carries information about the portion of space it covers. Points are identified with the pixels they lie in, while lines and regions are obtained as aggregations of pixels (digital lines and regions) on the basis of membership attributes. Therefore, spatial entities are not explicitly represented in a raster, while they can be obtained as relations that cluster pixels according to their common attribute. A raster model intrinsically defines a subdivision of the domain, while a collection can be obtained by using multiple members for pixels (i.e., each pixel may belong to more than one entity).

The raster representation is obviously space-oriented, and it provides direct access to information about a given location or extent of space. It has the disadvantage of providing an approximate geometry, whose accuracy is dependent on the resolution of the grid, but, on the other hand, its regular structure greatly helps organizing spatial information.

Not all operations are supported in both models (e.g., some systems compute buffer zones around vector features by converting to the raster domain and back). In general, raster models are more suitable to support queries *by location*, while vector models are more suitable to support queries *by content* [244]. As a consequence, many commercial systems adopt hybrid representations, which permit using either model depending on the specific task and data analyzed. Hybrid models involve the use of algorithms to convert from one model to the other.

Techniques used to store, process and analyze data in a raster model, as well as conversion algorithms, are more typical of digital geometry [165], and image processing [220, 234] than of classical Computational Geometry. Therefore, in the sequel we will focus on models and methods for vector models.

2.3 Subdivisions as Cell Complexes

Algebraic topology provides a well-established theory on subdivisions of a topological space into *cells*. In the mathematical literature there are different definitions of cell complexes, none of which is general enough to model regions and features as those manipulated in GISs. However, many models have been developed on the basis of classical cell complexes, and some efforts have been undertaken to extend classical theory to entities manipulated in a GIS.

Simplicial complexes are a special class of cell complexes with good combinatorial properties that make them easy to construct and manipulate. Simplicial complexes have been used in GISs for encoding generic subdivisions by triangulating each region and including all point and line features as vertices and edges [102, 291]. Although the approach is theoretically sound and elegant, it may not be practical: geographic data mainly contain large regions with complicate boundaries and features, and fragmentation into triangles causes a relevant increase in storage requirements. A similar approach, with similar drawbacks, consists of using complexes with convex cells, such as trapezoidal decompositions [94]. Regular cell complexes, in which regions are generic, simply-connected polygons without features, correspond to planar subdivisions commonly used in computational geometry. Regular cell complexes are suitable to model maps that do not contain point or lineal features.

The most general cell structure provided by algebraic topology is the CW-complex [183]: its Euclidean realization in two dimensions is a subdivision in which the interior of each region is homeomorphic to an open disk. Such structures admit some features, such as cuts incident into the generalized boundary of a region, and holes incident into such cuts. However, regions must be simply connected: punctures, generic cuts, and "islands" are not admitted. Each region can be fragmented into a minimal number of simply-connected components: such fragmentation usually yields only a modest increase in storage. Although some efforts have been undertaken to generalize CW-complexes to support GIS data [46, 218, 240, 292], no substantial progress in the direction of data structures and computational techniques has been made so far.

On a different perspective, some authors have proposed approaches based on integer geometry, motivated by robustness issues: points in a map, including endpoints and intersections of lines, are constrained to lie at the nodes of an integer grid. In particular, an algebra based on discrete generalized maps, called *realms*, has been specified and implemented in [128, 129].

2.4 Topological Data Structures

Data structures for encoding both entities and topological relations in a subdivision are usually referred to as *topological* data structures. The literature offers a number of data structures for encoding regular subdivisions of the plane or of a surface, which support efficient evaluation of spatial relations therein. According to Lienhardt [177] and Brisson [25], the models underlying data structures for cell complexes can be classified into:

- *explicit-cell representations*, where the entities encoded directly correspond to the cells of the complex;

- *implicit-cell representations*, where basic elements of a single type are used, on which element functions act, while cells are implicitly represented.

The incidence graph [79] is the prototypical data structure of the former class: nodes correspond to the cells (in two dimensions, vertices, edges, and faces), while arcs linking cells of different dimension correspond to incidence relations. The *symmetric structure* described in Woo [290] is an implementation of such graph for the boundary representation of solids.

Implicit-cell representations are composed of a set of basic elements (e.g., *darts*, *cell-tuples*, etc.) plus a set of functions acting on such elements (e.g., *involutions*, *switch operators*, etc.). Examples of data structures within such class are the *Doubly-Connected Edge List (DCEL)* by Preparata end Shamos [221], the *winged-edge* by Baumgart [16], the *half-edge structure* by Mäntylä [187], the *vertex-edge* and *face-edge structures* by Weiler [285], and, in a generic dimension d, the *cell-tuple structure* [25], and the *generalized maps* [177]. The *quad-edge structure* [123] can be viewed as a specialization of the cell-tuple structure in two dimensions (see Chapter 10 for a treatment of data structures for planar subdivisions).

Some special structures have been developed in the literature for triangulations and trapezoidal decompositions, which can exploit nice characteristics of such subdivisions. A widely used data structure for triangulations consists of storing, for each triangle, its three vertices and its three edge-adjacent triangles [172]. A similar data structure, called the *quad-view* data structure, has been proposed recently in the context of GIS to encode a trapezoidal map [94].

The data structures proposed to encode general subdivisions with features are basically extensions of incidence graphs to non-regular situations. For instance, data structures adopted in [58, 136, 198] are variations on the DCEL, which is extended to handle holes and features.

2.5 Multiresolution Data Structures

Multiresolution modeling offers interesting capabilities for spatial representation and reasoning in a GIS: from support to map generalization and automated cartography, to efficient browsing over large GISs, to structured solutions in wayfinding and planning.

Early approaches were based on a common practice in cartography, i.e., to have a collection of different maps of the same area at different scales, hence encoding information at different levels of detail and accuracy. Such approaches are often referred to in the GIS literature as *multiple representations*. Some research has also been undertaken, concerning assessment of consistency of different representations [87].

In [274] some hierarchical data structures, called *reactive*, are proposed, and some others are reviewed for multiresolution representation of lines and maps. The specific purpose of such structures is to allow one to retrieve a more or less detailed/accurate description of the objects. A description of the whole domain at a low level of detail is retrieved from a reactive model through a breadth-first traversal of the first levels of a tree, while a description of a portion of the domain at high level of detail is retrieved through a depth-first traversal. Examples of such structures are the reactive tree, based on the R-tree; the BLG-tree for line generalization; and the GAP-tree, to support generalization of area partitionings [275].

More recent approaches have tried to achieve multiresolution on the basis of an integrated data model, which can include information on maps at different levels of resolution together with explicit relations among them. The possibility of developing models that can support a multiresolution representation of maps through hierarchical structures based on trees of cells has been outlined in [107]. Independently, a first hierarchical model that is formally defined on a mathematical basis has been proposed in [19]: such model is described by a tree of maps at different resolutions, where each map is the refined description of a simple region of its parent node in the tree.

In [223], a more complete multiresolution model for maps represented by generalized subdivisions has been proposed, which deals with combinatorial and metric aspects of geographic maps separately, and formally relates all different representations of an entity through the various levels of resolution. The topological structure of a geographic map is completely captured by a purely combinatorial structure called an *abstract cell complex*. Map generalization, i.e., an operation that relates two consistent maps at different detail, is expressed by a continuous mapping between the abstract cell complexes representing the two maps (see also Section 3.3). Suitable rules permit to control such functions in order to guarantee that generalization occurs through gradual changes. The metric aspects concerning changes in accuracy can be controlled separately through the concept of line homotopy. The iterative application of simplification mappings satisfying both topological and metric constraints permits to define a sequence of generalized maps of the same area. This sequence provides means to organize the various maps, together with the mappings relating them, either in a multi-layered or in a tree structure.

3 Map data processing

3.1 Spatial Queries

Spatial queries are queries in a spatial database that can be answered on the basis of geometric information only, i.e., the spatial position and extent of the objects involved. A spatial query is defined by a *query space* S, i.e., either the whole spatial database, or a portion of it obtained through suitable filters; by a *query object* q that can either belong or not belong to the database; and by a *spatial relation* $\Re$. A generic query is thus defined as follows:

$$\textit{return all objects } s \in S \textit{ that are in relation } \Re \textit{ with } q.$$

A classification of spatial queries follows directly from the classification of spatial relations (see Section 2.1).

Computational techniques that can be adopted to answer spatial queries depend on the nature of the query space (and on the model which encodes it), on the nature of the query object, and on the nature of the relation. However, a simple exhaustive study of all possibilities shows that most spatial queries can be eventually reduced to few basic problems in computational geometry.

In [58] a study is presented on the formalization and solution of topological and metric queries on geographic maps encoded through a cell complex. It is shown that topological queries are solved by map traversal when the query object belongs to the map, while they are always reduced to point location, line intersection, or polygon intersection, when the query object does not belong to the map (see Chapter 10 for a survey).

An efficient algorithm has been proposed in [53], which is specific for map traversal in GISs, and provides the basis for the solution of several topological queries on a map. This algorithm traverses a planar subdivision and reports all its regions, without making use of mark bits in the data structure or a stack. Thus, it is especially suitable for a map stored in a read-only GIS database.

On the contrary, metric queries are not directly supported by the topological structure of the map, and need auxiliary structures - such as spatial indexes, or Voronoi diagrams - to be answered efficiently. The use of Voronoi diagrams [11] for answering spatial queries in GIS has been discussed by Gold and Roos [114, 115, 116, 117]. The Voronoi diagram is seen as an intermediate model between the raster and the vector models. As observed by Gold in [116], "the Voronoi model has the complete tiling property of the raster model, but the tiles are based on the proximal region around each object". Both static and dynamic algorithms for constructing a Voronoi diagram of 0-, 1- and 2-dimensional entities in the plane have been developed in the computational geometry literature (see Chapter 5). Gold and Roos show that the Voronoi diagram is a persistent, locally modifiable data structure suitable for algorithms like navigation queries, buffer zone generation, polygon labeling, intersection of a map with a line segment, interpolation of a surface at a new point [115, 117], etc.

Techniques based on spatial indexes are treated in Chapter 17. The interested reader can also refer to the two books by Samet [242, 243].

Queries on collections of entities can benefit by less support from relations provided explicitly by data structures. In principle, computational geometry algorithms working on arrangements may give solutions to some such queries, but such algorithms are not applied in practice, perhaps because they are difficult to code, and have an intrinsically high computational complexity.

Popular techniques for processing collections of entities are based on spatial indexes, which are simple to code, and perform well in the average case. In [97] a thorough analysis of the average case complexity in accessing quadtrees is presented. In [8, 140, 141], several algorithms for spatial queries based on spatial indexes are presented. Refer to Chapter 17 and [243, 274] for more details on searching techniques based on spatial indexes.

Spatial queries can also be combined with queries related to non-geometric attributes of objects. In this case, a rigorous analysis should take into account both geometric and non-geometric issues. The efficient treatment of such hybrid queries, and the development of suitable integrated models and techniques, as well as of methods for analyzing their performances, are hot open issues that require interdisciplinary research.

3.2 Map Overlay

It is common practice in GISs to organize information into layers (e.g., land use, hydrography, road network, etc.), and to produce maps by overlaying layers of interest. The overlay process is the basic tool for solving complex spatial queries (the so-called *spatial join*). In such queries, the input maps are not only overlapped, but also combined on the basis of their attributes (e.g., by using boolean expressions). Here, we deal only with the geometric aspect of map overlay, where techniques from computational geometry can be applied.

Overlay of maps in raster format reduces to combining the attribute values of corresponding pixels and does not involve computational geometry techniques. Conventional trasformations are used to align raster grids having different orientations or cell sizes [98]. Here, we focus on the overlay of vector maps.

In GIS, there are several cases of overlay, such as overlay between two subdivisions (also called area-to-area overlays), line-to-area overlays, point-to-area overlays. Moreover, both in area-to-area, and in line-to-area overlays, input datasets may be collections of possibly intersecting regions, and lines. Two maps may be combined to intersect, form a union, or one map may be used to update another; multiple maps may be combined using Boolean expressions.

On the other hand, in computational geometry, overlay problems have been studied in a much more standard form:

- *segment intersection* problem: find all the intersection points within an arbitrary set of segments;
- *red-blue intersection* problem: find the intersection points of two sets of segments, where no intersections exist within the same set;
- *superimposing plane subdivisions*: given two subdivisions Σ_1 and Σ_2, compute a subdivision Σ' whose edges and regions are obtained as the intersection between edges and regions of Σ_1 and of Σ_2.

Segment intersection algorithms can be used to overlap line maps (e.g., road networks). Red-blue algorithms apply to map layers consisting of general subdivisions with features (see Section 2.4). Some algorithms for plane subdivisions have further restrictions, for instance, to trapezoidal maps [95], to convex regions (e.g., a triangulated map) [207]).

In all overlay problems, the number of intersections is $k = \Theta(n^2)$ in the worst case, where n is the input size. The problem complexity is bounded by $O(n+k)$. A brute force solution consists of computing the intersections between every pair of entities from the two given layers, and works in $\Theta(n^2)$.

Methods developed in the GIS community mainly consist of heuristic "filters" which try to reduce the number of tested pairs by superimposing a hierarchical spatial index. The filter is based on a decomposition of the plane into blocks, and on distributing the objects to be intersected (the edges and regions of each layer) among the various blocks. Pairs of objects which lie in disjoint blocks of space can be discarded from consideration since they cannot intersect. Then, an exhaustive search for intersections is applied within each block. The filter, however, is not guaranteed to be effective, i.e., some redundant tests are still possible. Nevertheless, these techniques show a good practical behavior. Even if they turn into an exhaustive intersection search in the worst case, their efficiency has been widely experimented in practice. For some methods, a theoretical analysis of the average randomized case is available (e.g., see [93]).

Filtering methods can be subdivided into (i) methods decomposing the space into *disjoint blocks*, evenually splitting objects that cross several blocks; and (ii) methods decomposing space into *possibly overlapping blocks*, storing each object in exactly one block.

The simplest decomposition scheme based on disjoint blocks is a *uniform grid* [104, 105]. The resolution of the grid determines the efficiency of the filter. Uniform grids show poor performance when data are irregularily distributed.

Quadtrees [242, 243, 210] are adaptive grids, based on a recursive partition of an initial square into four sectors. In this way, a different grid resolution can be achieved in each part of the domain according to the local density of data. Many variants of quadtree-like structures have been proposed, depending on the type of data to be stored: either the regions or the edges of a map. Boolean operations are easily done on maps with a superimposed quadtree-like structure. The two input quadtrees are traversed synchronously, by visiting nodes corresponding to the same portion of space at the same time; a quadtree for the result of the boolean operation is also incrementally built (see [243]).

Binary space partitioning (BSP) is a technique born in the image processing field and lately applied to GIS [273]. Still based on a recursive subdivision of space, it uses a more flexible scheme. At each step, the current block is subdivided into two parts by an arbitrary straight line. The resulting structure can be described as a binary tree, whose nodes correspond to convex polygons. BSP-trees support algorithms for performing boolean operations on objects stored in them [205].

Methods based on overlapping blocks filter candidate pairs by comparing bounding boxes. The *bounding box* of a two-dimensional entity is the smallest axis-parallel rectangle that completely contains it.

Structures of the *R-tree* family [127, 235, 250, 17, 92] consist of trees, with degree bounded by a predefined constant value, where data objects are stored at the leaves and every internal node represents the bounding box of all objects stored in its children. The bounding boxes associated with different nodes (even within the same layer) may overlap. A separate R-tree is built for any map layer. The intersections between the two layers are computed while descending the trees: at each step, tests on the bounding boxes are performed, and the results of such tests are used to cut off some branches of the trees, which are guaranteed to produce no intersection.

Intuitively, an 'ideal" R-tree should be balanced, its bounding boxes should as disjoint as possible, and the amount of empty space enclosed in them should be low. In fact, these properties enhance the efficiency of the filter. The numerous existing variants of R-trees are characterized by different criteria used to evaluate the 'good quality" of an R-tree, and by different construction strategies. Common criteria try to minimize area covered by the rectangles [127], or to avoid overlapping rectangles [250], or to aggregate objects based on their spatial proximity [235, 92]. Since many R-trees support dynamic insertions and deletions of objects, there is a trade-off between update time and the quality of the tree produced. R*-trees [17] seem to be the most efficient in practice.

In computational geometry, the purpose has been to find algorithms with a good theoretical complexity [18, 32, 30, 31, 41, 124, 207]. Some of such techniques, especially the plane sweep paradigm (see below) have been integrated in GIS environments [166, 167]. A classification of the major existing approaches can be as follows:

- an approach based on a *segment tree*, i.e., a special data structure which hierarchically partitions the plane according to the configuration of input segments [213, 31];
- a *randomized incremental* approach [41, 22], which computes all the intersections generated by a generic set of segments by adding the segments one at a time;
- a *sweep-line* approach, which finds the intersections while translating a line through the plane and monitoring the segments intersected by such a line [18, 30, 32, 207];
- an algorithm based on *topological sweep*, i.e., a sweeping process driven by the topological structure of the input subdivisions [95, 124].

The problem of overlapping subdivisions encoded through hierarchical models is faced in [184].

A segment tree is a binary tree whose leaves represent, in left-to-right order, the vertical strips defined by every pair of consecutive vertical lines drawn through the segment endpoints. Data segments are associated with nodes of the tree in an appropriate way. Then, it is sufficient to test intersections only between segments associated with the same node. Space requirements are reduced by dynamically constructing only a portion of the tree at a time: a single root-to-leaf path [31], or a single level [213].

In incremental algorithms, segment intersections are reported while constructing a trapezoidal decomposition of the plane. The *trapezoidal decomposition* induced by a set $\mathcal{S}$ of segments is obtained by drawing a vertical line through each segment endpoint or intersection point, until another segment of $\mathcal{S}$ is encountered in both directions. Efficiency in locating the trapezoids of the current decomposition affected by the insertion of a new segment is obtained by means of special auxiliary data structures which encode the history of the construction [41, 22].

Sweep-line algorithms are evolutions of the classical sweep-line method of Bentley and Ottman [18] for reporting the intersections in a set of segments. The basic idea is moving a vertical line left-to-right on the plane, and monitoring the segments intersected by such a line. The key observation is that, if two segments intersect, they must appear in consecutive positions along the line at some instant. Specialized sweep-line algorithms have been defined by relaxing the straightness condition of the line and/or the left-to-ritht order in intersection reporting; also, some algorithms require additional properties on the input segments [31, 32, 207].

The topological sweeping [95, 124] exploits the topological relations among the edges of a plane subdivision in order to advance the sweepline. In practice, the sweep-line is replaced by a cut on the graph induced by the edges of the two input maps and their intersection points, which is progressively moved letft-to-right.

Algorithms developed in computational geometry are summarized in Table 1. All the above approaches have a computational complexity of $O(n \log n + k)$, where n is the size of the two maps and k is the output size (for randomized algorithms this is an expected time complexity). The only exception is the topological sweep method which leads to $\Theta(n + k)$ worst-case optimal algorithms.

Andrews et al. [7] provide an experimental comparison of computational geometry overlay algoritms and GIS overlay algorithms. The compared approaches are the uniform grid [104, 105], the quadtree [242, 243], the BSP tree [273] from the GIS literature, and the Bentley and Ottman's sweep [18], the trapezoidal sweep [32], and the hereditary segment tree [213] from the computational geometry literature. The results show that filtering heuristics typically perform better because they take advantage of the typical characteristics of GIS data (short segments with a low number of intersections per segment). On large datasets, however, the grid tends to increase running time, and quadtrees and BSP trees may introduce a serious space overhead on large datasets. Asymptotically efficient computational geometry methods do not perform well on GIS data. An exception is the algorithm by Chan [32], which is competitive with GIS-based approaches, even on large data sets. The segment tree algorithm is the best one on artificial datasets, but the worst one on real GIS data.

Algorithm	**Approach**	**Input**	**Time**
[22, 41]	Incremental	Segments	$O(n \log n + k)$
[31, 213]	Segment tree	Red-blue segments	$O(n \log n + k)$
[18]	Sweep-line	Segments	$O((n + k) \log n)$
[207]	Sweep-line	Plane subdivisions with convex regions	$O(n \log n + k)$
[32]	Sweep-line	Red-blue segments	$O(n \log n + k)$
[30]	Sweep-line	Segments	$O(n \log n + k)$
[95, 124]	Topological sweep	Plane subdivisions	$\Theta(n + k)$

Table 1: Major map overlay algorithms. For each algorithm, the worst-case time complexity and the input requirements are indicated. Parameters n and k denote the input and the output size, respectively; the time complexity for incremental algorithms has been computed through a randomized analysis.

3.3 Geometric Problems in Map Generalization

Map generalization is known in cartography as a process of selection, and representation of detail to be shown on a map at a given scale. Map generalization involves different tasks, such as removing objects, simplifying geometry, aggregating several objects into one, etc. In GIS, generalization is more generally intended as a reduction of "information density" in a geographic database, while preserving its overall structure and semantics.

Generalization is a complex task, both conceptually and computationally. It involves geometric aspects of geographic entities (shape, structure, and detail), as well as non-geometric ones (role and relevance in the map context). In traditional cartography, generalization processes are classified into the following basic operations [232]:

- *selection* decides whether an entity should appear (i.e., be explicitly represented) or not in the simplified map;
- *aggregation* merges different entities into a single one;
- *collapse* reduces the dimensionality of an entity (e.g., a small region collapses to a point, a thin region collapses to a line);
- *symbolization* transforms a geometric entity into a symbol;
- *simplification* reduces the accuracy in representing the shape of an entity;
- *exaggeration* modifies the shape of an entity to highlight features that would become hardly visible at small scale;
- *displacement* also modifies the shape to set apart entities that would become too close to each other.

Selection and aggregation are usually controlled through non-geometric rules. Symbolization is also non-geometric, except for the problem of avoiding overlapping symbols, which is discussed separately in Section 3.4. The remaining operations are primarily geometric, since they affect the shape of entities explicitly represented in the map at reduced resolution.

Simplification is by far the most studied problem in generalization. The best known and used algorithm is probably an early refinement heuristic proposed by Douglas and Peucker for a simple open polygonal chain [72]. Starting from an initial approximating sequence, the algorithm inserts at each cycle the point maximizing the distance from the current approximation, until the maximum error gets below a given threshold. A straightforward implementation runs in $O(n^2)$ time in the worst case; a sophisticated implementation running in $O(n \log^* n)$ time has been proposed recently [139]. The GIS literature is full of other heuristics proposed for the same purpose, which are usually compared empirically (see [192] for a survey and experimantal comparisons; [176, 214] for more recent examples).

A rigorous approach to the problem is due to Imai and Iri [149] who first gave a formalization of line simplification as an optimization problem. Namely, given an input curve and some metric to measure the error in approximating it, they outline two basic problems: (i) minimizing the number of vertices of the output chain for a given threshold error; (ii) minimizing the error of the output chain for a given number of vertices. Imai and Iri give algorithms, and list results by other authors to cover both problems under four different metrics. Other results are also referred to by Guibas et al. [125]. All algorithms have superquadratic time complexity for generic polygonal chains, but for a special metric, a subquadratic time complexity can be achieved [147]. Problem (i) can be solved in linear time for the special case in which the input is the image of a piecewise-linear function [148].

An elegant definition of problem (i) is also suggested by Guibas et al. [125]: a homotopy class is obtained by convolving the input line with a disk having a radius equal to the given threshold. The solution is a polygonal chain with minimum links, which is homotopic to the input line into such region. Linear algorithms for computing homotopy paths inside simple polygons were proposed by Suri [258], Ghosh [113], and Hershberger and Snoeyink [138]. All such algorithms follow a greedy approach. Guibas et al. also study problem (i) as a problem of ordered stabbing [125]. Only disks centered at vertices of the input chain are considered: the solution must be a minimum-link chain stabbing such discs in order. For such a problem, they give an $O(n \log n)$ suboptimal greedy algorithm, as well as an optimal quadratic algorithm based on dynamic programming; finally, they give a linear time optimal solution for the special case in which discs in input are all disjoint.

All algorithms cited above cannot be considered as ultimate solutions to the problem, because they cannot guarantee that the output will be free of self-crossings. A negative result by Guibas at al. [125] shows that the problem of finding a minimum-link simple polygon for a given homotopy type is NP-hard. In the same paper, the problem of line simplification is also studied in the context of a map: given a plane straight-line graph induced by a set of non-crossing polygonal chains, find a minimum-link approximation (in the homotopy class of the map, as outlined above for lines). Also this problem is NP-hard: the difficult part is in positioning vertices that have more than two incident edges.

The more general problem of subdivision simplification have been recently faced by Weibel [287] and de Berg et al. [51]. Weibel [287] defines constraints on the simplification process based on cartographics principles, classified into metric, topological, semantic (like preservation of class membership) and gestalt principles (constraints used by cartographers). de Berg et al. [51] face the problem of simplification with the main purpose of reducing the complexity of a subdivision. They propose a nearly quadratic algorithm for the following problem: given a polygonal line, a set of feature points, and a real number $\varepsilon > 0$, compute a simplification that guarantees (i) a maximum error ε, (ii) that feature points remain on the same side of the simplified chain as of the original chain, and (iii) that the simplified chain has no self-intersection.

Relatively little research has been conducted on merging of areal features. The earlier techniques were usually raster-based involving mathematical morphology operators such as dilation and erosion. Collapse problems, like reduction either of a ribbon-shaped area to a center line, or of an areal feature to a point, are faced by using the medial axis transform (see [281] for a survey, and [74, 160] for algorithms on medial axis). Jones et al. [154, 281] describe an implementation of a few generalization operators based on a triangulated spatial model of a map built on a constrained Delaunay triangulation of the map vertices and edges (see [173, 248]). Such operators allow detection of conflicts and their resolution, dimensionality reduction through skeleton generation, boundary simplification and merging of nearby objects.

On a more general perspective, Saalfeld [241] recently suggested that the basic operations of map generalization can be expressed in terms of graph drawing. Although such an approach seems to be promising, so far it is just a challenge launched to experts in graph drawing, while no algorithms based on it have been proposed yet. Another approach to map generalization has been proposed recently by Puppo and Dettori [68, 223], which is based on functions defined between cell complexes (describing maps) and continuous under the finite topology. This approach is mainly intended to give consistency control on operations that affect the topology of the map, and to the definition of multiresolution models. By following the approach by Puppo and Dettori, algorithms for geometric consistency have been recently proposed by Delis and Hadzilacos [67].

To summarize, we remark that while only empirical or manually-assisted solutions to map generalization are used in practice, only partial solutions to a few theoretical subproblems have been proposed. A unifying framework that can formally incorporate all operations involved in map generalization is still an open issue.

3.4 Map Labeling

Map labeling is a problem of critical importance in cartography. It consists of positioning labels on the map under the following constraints: they must be legibly referred to spatial entities; they cannot overlap; they cannot cover relevant features of the map. In manual cartography, this task is estimated to take about 50% of the time to draw a map; hence the importance of finding automatic procedures to place labels.

Map labeling is often subdivided into three subproblems, namely labeling areas (e.g., countries, lakes), labeling lineal features (e.g., rivers, roads), and labeling points (e.g., cities). According to some authors [70], there exists a hierarchical relationship that permits to solve such problems separately: area features are placed before point features, which in turn come before line features. However, most literature has been devoted to studying point labeling.

Informally, point feature labeling is defined as follows: given a set of points spatially located, and a set of labels referred to each point, find the choice of label positions that minimizes the total number of label-label and label-point overlap. Labels are modeled as rectangles of variable size. Most cartographers agree on constraining label to have the referred point either at one of its four corners, or at one of the midpoints of its four edges.

Many solutions to point labeling have been based on greedy techniques with backtracking to escape invalid configurations (see, e.g., [70, 152] and references therein). Different heuristics were adopted to speedup algorithms, which however have exponential time complexity in the worst case. Cromley [47] and Zoraster [296] independently proposed an approach based on integer programming and iterative solutions based on relaxation. Christensen et al. [38] give an extensive survey and an empirical analysis of several computational techniques for map labeling. Moreover, they propose an algorithm based on gradient descent and another one based on simulated annealing. Both such algorithms are based on iteration: starting from an arbitrary configuration, they make local improvements. With the gradient descent, the local change that gives the best improvement is performed at each iteration. However, this method can get stuck at local minima without achieving the optimal solution. In the algorithm based on simulated annealing, local minima are avoided through a stochastic approach. At each iteration, local random changes are performed, which are discarded with increasing probability if they worsen the cost of the configuration. Christensen et al. show how to manage the cost to include not only the number of conflicts, but also placement preferences, and possibly deletions of labels in excess to escape from illegal configurations.

Formann and Wagner [99] have studied a version of point labeling, where only four positions per label are allowed; they have shown that the problem is NP-complete through reduction to 3-satisfiability. They also propose an optimal algorithm running in $O(n \log^2 n)$ time for arbitrary rectangles, and in $O(n \log n)$ time for equally-sized squares, when only two possible positions per label are allowed. On this basis, they propose an approximate solution for the case of four possible positions that can be found in $O(n \log n)$

time and ensures correct placement of labels whose size is at least 50% of the optimal. They also show that this is the best approximation ratio that can be achieved in polynomial time for this specific problem.

Kučera et al. [168] studied the same problem, but developed an exact super polynomial algorithm that can be applied for sets with up to about 100 points. Wagner and Wolff [279, 280] have noted that in practice the approach of Formann and Wagner hardly ever results in label sizes significantly larger than half the optimum. They study variations of the problem and find a practical way to improve the size of the squares. Recently, Agarwal et al. [4] investigate the problem of computing a large non-intersecting subset in a set of rectangles in the plane.

An exhaustive bibliography on the map labeling problem can be found at http://www.inf.fu-berlin.de/map-labeling/papers.html.

3.5 Other Analysis Issues

In this subsection, we will deal with other issues that have received some attention in both computational geometry, and GIS: namely, map conflation, path planning, and construction of cartograms. The path problem in the context of terrain models will be discussed in Subsection 5.4. Refer also to Chapter 15 for further details.

Maps provide approximations of the real locations and extents of geographic objects. The representation of the same object may differ in two maps for small changes in its shape and displacement. Map conflation is an operation aimed to reconcile two variants of a map of the same area, in order to compile a map that integrate information from both [238].

The dissertation of Saalfeld [241] provides both a mathematical modeling framework, and a collection of algorithms for building a conflation system. Basic issues in this context are plane graph matching, and geometric transformations on planar regions. Plane graph matching does not involve graph isomorphism: heuristics for feature matching, and neighborhood relations are used. Geometric transformations and final merging of the two maps is performed by using a triangulation of the maps [237].

The computation of optimal paths has been studied in GIS mainly for road networks. In these cases, problems can be modeled in a graph-theoretical setting, and generally do not involve geometric computation. Such methods cannot be applied, for instance, in planning a new road through a region: the road will not, in general, follow an existing edge of the map, but rather it will traverse the interior of the region along a path that did not exist before. Path computation involves geometric calculations, and cannot be modeled as a discrete graph theoretic problem (see the work of Mitchell and Suri [197] for a survey).

The *Weighted Region Problem* consists of finding an optimal path in a planar subdivision, where a non-negative cost is associated with each edge and region. Such cost is the cost of traveling one unit of lenght across the region/along the edge, and may be based on soil, vegetation, etc. Costs can also be infinite, and thus model obstacles which cannot be traversed. The problem is relevant both to motion, and to planning situations such as building highways, railways, pipelines, etc. The weighted region problem has been intensively studied by Mitchell. In [196] an algorithm is proposed that finds an approximated shortest weighted cost path from a fixed point in $O(n^8 \log n)$, where n is the size of the problem. In a recent paper [189], Mata and Mitchell propose an algorithm that can deal for an arbitrary pair of points, and has an improved time complexity (an n^3 factor instead on n^8). Such an algorithm has been successfully implemented.

An $O(n^2)$ algorithm for the special instance of the above problem in which only regions with weights in the set $\{0, 1, +\infty\}$ is described in [112]. In [266] algorithms for the weighted region path problem working on a raster model as well as an implementation of the algorithm of Mitchell and Papadimitriou based on constrained Delaunay triangulation (see [173]) are described and compared.

A *cartogram* is a geographic map that is purposely distorted so its spatial properties represent quantities not directly associated with position on the globe [71]. Distorsions should act as homeomorphisms, i.e., continuity of the map should be preserved. Some automatic methods for generating cartograms have been developed that use iterative techniques for computing differential equations involving force fields [71, 263]. Recently, Edelsbrunner and Waupotitsch [82] have proposed a combinatorial algorithm that is based on a sequence of local piecewise linear homeomorphic changes, acting on a tiling of the map made of regular triangles.

4 Terrain Data Modeling and Processing

Terrain processing requires input, data models, and algorithms different from those needed for 2D data. Terrain processing is also a major source of geometric problems, which stimulate the use of computational geometry, both in applying existing techniques, and in developing new ones.

The term *Digital Terrain Model* (DTM) is used generically to denote a model representing terrain relief on the basis of a finite number of samples [193]. Elevation data are acquired either through sampling technologies (on-site measurements or remote sensing: tacheometers, photogrammetry, SAR, etc.), or through digitization of existing contour maps [215]. Raw data come in the form of elevations at a set of points, either regularly distributed, or scattered on a two-dimensional domain; chains of points may form polygonal lines, approximating either lineal features, or contour lines.

On the basis of sampled data, different models can be built: contour maps, rasters, and mathematical surface representations. Mathematical models are either global (i.e., defined through a single function interpolating all data), or local (i.e., piecewise defined on a partition of the domain into patches). Regularly distributed data generally lead to regular square grids, while scattered data generally lead to triangulated irregular networks. Since we are concentrating on models and methods rooted in computational geometry, we will not cover raster models, and digital geometry techniques to process them, as well as global mathematical models, and algebraic techniques.

In the rest of this section, we will deal essentially with general problems related with construction and storage of DTMs, while more specific problems regarding their analysis will be described in the next section. The interested reader can also refer to a survey by van Kreveld [270] for more detailed issues about DTMs, and in particular algorithms on TINs, and to a survey by Magillo and Puppo [185] for issues on parallel algorithms for terrain processing.

4.1 Classical Terrain Models

A *topographic surface* (or *terrain*) σ is the image of a real bivariate function f defined over a compact and connected domain Ω in the Euclidean plane, i.e., $\sigma = \{(x, y, f(x, y)),\ (x, y) \in \Omega\}$. Given a real value q, the set $C_\sigma(q) = \{(x, y) \in \Omega,\ f(x, y) = q\}$ is the set of *contours* of σ at height q, and it is formed by a set of simple lines (provided that f has no relative maxima, minima, saddles, or horizontal plateaus at height q).

A *Digital Terrain Model (DTM)* is a model providing information on such a surface on the basis of a finite set of data. Terrain data are measures of elevation at a set of points $V = \{v_0, \ldots, v_N\} \subset \Omega$, plus possibly a set of non-crossing straight-line segments $E = \{e_0, \ldots, e_M\}$ having their endpoints in V. Data points in V can either be scattered or form a regular grid. Lines in E can possibly form a collection of polygonal chains.

Three classes of DTMs are usually considered in the context of GIS [216]:

- *Polyhedral terrains*: a polyhedral terrain is the image of a piecewise-linear function f. A polyhedral terrain model can be described on the basis of a partition of the domain Ω into polygonal regions having their vertices in V (and such that the segments of E appear as borders of regions). The image of f over each region is a planar patch. The most commonly used polyhedral terrain models are *Triangulated Irregular Networks (TINs)*, in which all regions are triangles.

- *Gridded elevation models*: a gridded elevation model is defined by a domain partition into regular polygons induced by a regular grid over Ω. The most commonly used gridded model is the *Regular Square Grid (RSG)*, in which all regions are squares. Function f is usually a bilinear function interpolating the vertices of the grid. In some cases, an RSG is treated as a raster, and a constant function is used over each region. This is said a *stepped model*, and it is obviously non-continuous across edges of the grid.

- *Contour maps*: given a sequence $Q = \{q_0, \ldots, q_h\}$ of real values, a *digital contour map* of σ is an approximation of the collection of contours $C_{\sigma,Q} = \{C_\sigma(q_i),\ i = 0, \ldots, h\}$. Contours in digital contour maps are often represented as point sequences; a line interpolating points of a contour can be obtained in different ways: from the simplest case of a polygonal chain to spline curves of various orders.

Contour maps are easily transposed onto paper and best understood by humans, but are not suitable for performing complex computer-aided terrain analysis. This is due to the complete lack of information about terrain morphology between two contour lines. Automated terrain analysis is performed on RSGs or TINs. As for maps, some operations are better supported by RGSs, and others by TINs. Advantages and disadvantages of the two models are illustrated in [27, 9].

Triangulated Irregular Networks are the most interesting models for application of computational geometry techniques. A TIN is based on a triangulation T of the domain with vertices at the projection on the x-y plane of the data points (or of a subset of them) and sometimes including a set of straight-line segments as subset of the edges of T. The resulting piecewise-linear surface interpolates data elevations at all points and lines whose projections are vertices and edges of the triangulation, respectively. Data structures describing a TIN are those for encoding triangulations in the plane (see Section 2.4).

There exists also a flourishing literature on TINs using smooth patches at each triangle, in order to achieve C^1 continuity of the resulting surface [206]. Building smooth patches is easy, though expensive because of numerical computation involved, if surface normals are provided (or estimated) at each datum.

There are different strategies to define a TIN having its vertices at a given set of data. In general, the connection topology of vertices is required to satisfy some optimality criterion. Optimality criteria can be defined either for the underlying triangulation, or for the surface itself. A measure can be assigned to each triangle of a triangulation (e.g., its area, the minimum/maximum/mean size of its edges, or of its angles). Then, the key for comparing triangulations can be the sum, or the maximum, or the minimum, or a lexicographically ordered vector of the measures of all triangles. The most commonly used triangulation is the Delaunay triangulation which is optimal according to the following criteria: it maximizes the minimum angle of its triangles [172, 221], it minimizes the maximum circumcircle [48], and it minimizes the maximum containing circle [227].

Optimality criteria defined on the approximating surface have also been considered in the context of scattered data interpolation. Triangulations defined through such criteria are referred to as *data-dependent* triangulations, since optimization depends also on the z-values of data points, rather than simply on their x-y coordinates. Dyn, Levin, and Rippa [78] consider optimality criteria that can be defined through some cost function of the resulting surface. Examples of optimality criteria are the following:

- *three-dimensional criteria*: criteria like the max-min angle, the min-max angle, or the minimum edge length are considered on triangles and edges of the surface, rather than on their projections on the plane;

- *nearly C^1 criteria*: the piecewise-linear surface that is as close as possible to a C^1 approximation on the edges of the triangulation is selected;

- *variational criteria*: the unknown surface is assumed as the one minimizing some functional defined over a suitable space of functions. One example of such functional is the *roughness*, defined by the Sobolev semi-norm of the function. Surprisingly, the surface with minimum roughness is always given by the Delaunay triangulation of projected points, independently of the z-values of its vertices [230].

When the data set also includes a set of line segments, it is also important that such segments appear as either edges, or chains of edges in the TIN. In this case, either constrained [173, 248] or conforming [83, 239] triangulations are used. A *constrained triangulation* T of a set V of points and a set E of straight-line segments is a triangulation of V that contains E as a subset of its edges. A constrained Delaunay triangulation is a constrained triangulation satisfying a weaker version of the circumcircle property. A *conforming triangulation* T of a set V of points, and a set E of straight-line segments is a triangulation of V such that every segment of E may be decomposed into a chain of edges of T. In other words, the constraint segments are broken into pieces in a conforming triangulation in order to avoid long and thin triangles. See Chapter 5 for details. For conforming and constrained triangulations, only optimality criteria defined on the underlying triangulation have been considered so far. A survey of Delaunay triangulations (both non-constrained and constrained) for TIN creation is provided in [264].

4.2 Construction and Conversion Algorithms

Raw data can come in the form of a set of points V, and possibly of a set of lines E. In case the points of V are distributed regularly, an RSG is implicitly provided. Since sometimes DTMs are derived from digitizing existing contour maps, contours may also play the role of raw data. Hence, we have the following possible construction and conversion problems. Note that a clear distinction between conversion and construction techniques cannot be done since sometimes models and raw data are the same (e.g., regular distributed data points and RSGs).

1. **RSG from sparse points**. There are three possible approaches for constructing an RSG from a set V of scattered points [216]:
   (a) *pointwise methods*: the elevation at each grid point p is estimated on the basis of a subset of data that are neighbors of p. There are different criteria to define the neighbors that must be considered, e.g.: the closest k points, for some fixed k; all points inside a given circle centered at p, and of some given radius; the neighbors of p in the Voronoi diagram of $V \cup \{p\}$; etc. The basic geometric structure for all such tasks is the Voronoi diagram of the given data points. If raw data include line segments, then a Voronoi diagram of segments, or a bounded Voronoi diagram [248] can be used.
   (b) *global interpolation methods*: a unique interpolating function (usually, a polynomial) is computed, which interpolates elevations at all points of V. The RSG is obtained by sampling such function at grid nodes.
   (c) *patchwise interpolation methods*: the domain is subdivided into a number of patches, which can be either disjoint or partially overlapping, and either of regular or irregular shape. The terrain is approximated first within each patch through a function that depends only on data inside the patch. The elevation at grid nodes inside each patch is estimated by sampling the corresponding function.
2. **RSG from TIN**. Some systems first compute a TIN from sparse data points (and, possibly, lines), then they convert such a representation into an RSG [284]. This conversion is indeed a special case of patch interpolation methods described above. After computing the TIN, elevations at RSG nodes can be evaluated efficiently through scan-line conversion algorithms that run in linear time [98].
3. **RSG from contours**. Early methods performed this conversion as follows: a number of straight lines at given directions (at least a horizontal and a vertical line) are drawn through each node of

the grid, and their intersections with the contour lines are computed; terrain profiles along each line are approximated by some function interpolating contours at intersection points; the elevation at the grid node is computed as an average of its approximate elevations along the various profiles. The underlying geometric problem is to find intersections between the contour map, and the lines through each grid node: red-blue intersection algorithms can be used to this purpose (see Section 3.2). More recent approaches perform this conversion in two steps: contours are first converted into a TIN, then such a TIN is scan-converted to produce the final RSG [216].

   Carrara et al. [29] provide an experimental analysis of four different methods for deriving an RSG from contour lines; results on RSG's accuracy in preserving a coastline are reported in [28]. Eklund and Mårtensson [88] compare the accuracy of RGSs generated from contour lines and from scattered points.

4. **TIN from points**. A TIN is obtained from sparse data points by computing a triangulation having vertices at data points. In case raw data also include line segments, a constrained or a conforming triangulation is computed. Thus, the basis for TIN construction are algorithms for building a Delaunay triangulation, a data-dependent triangulation [78, 230], a constrained Delaunay [173] and a conforming Delaunay triangulation [83, 239]. The interested reader is referred to Chapter 5, and to [11] for further details.

5. **TIN from RSG**. This conversion is usually aimed at data compression: the adaptivity of the TIN to surface characteristics is exploited to produce a model of terrain that can be described on the basis of a reduced subset of elevation data from an input RSG. Hence, RSG to TIN conversion involves approximation. The construction of approximated TINs is treated in detail in Subsection 4.3.

6. **TIN from contours**. A TIN conforming to a given contour map should be based on a triangulation that conforms to the set of contours. This problem has been studied in the literature both in the context of GIS, and more generally in the reconstruction of three-dimensional object models.

   A constrained Delaunay triangulation, using projections of the contours on the x-y plane as input constraints, gives a conforming TIN, but it may produce artifacts, due to triangles having all vertices on the same contour line. Many algorithms adopt heuristics to avoid artifacts. In [37], flat triangles are avoided by adding the medial axis of each pair of adjacent contour lines to the dataset before computing the triangulation. Specific algorithms developed for TINs are described in [38, 110]. More general algorithm for 3D objects produce a triangulated surface from a sequence of cross-sections, resulting from the intersection of the object with a collection of parallel planes [159, 109, 111, 255].

7. **Contours from points**. As for *RSG from contours*, this conversion is usually done in two steps: an RSG or a TIN is built first from raw data, then contours are extracted from such a model.

8. **Contours from RSG**. If the RSG is a bilinear model, all square regions can be processed independently: possible intersections of the four edges of each region with contour lines are found in constant time. Contour segments inside each region are obtained by connecting intersection points at corresponding elevations. In a second step, corresponding contour segments from adjacent regions must be sewn together to form contour lines through a contour following technique.

9. **Contours from TIN**. Building contours from a TIN is completely similar to constructing contours from a bilinear model. Contour segments for triangular regions can be found independently while contour extraction needs contour following. In [13, 180, 269] efficient structures based on interval trees [269], kd-trees [180], and segment trees [13] are proposed, which permit to extract contours from a TIN in $\Theta(\log n + k)$ or $O(\sqrt{n} + k)$ time, where n is the size of the TIN, and k is the size of the resulting contour map. The search structure, however, can be a serious storage overhead. van Kreveld et al. [272] reduce the storage requirements by observing that a contour can be traced directly if one mesh element through which the contour passes is known. They give an algorithm which constructs the so called *contour tree*, a tree that captures the contour topology of a TIN, or an RGS, in $O(n \log n)$ with less additional storage than a previous algorithm proposed by de Berg and van Kreveld [54].

The problem of efficiently transmitting and compressing a TIN has been recently addressed by Snoeyink and van Kreveld [256] for the case of TINs based on Delaunay triangulations. Given a 2D Delaunay triangulation, they show that one can determine a permutation of the data in $O(n)$ time such that the Delaunay triangulation can be reconstructed from the permuted data in $O(n)$ time. More general methods for compressing triangular meshes (not just TINs), which exploit the compactness of *triangle strips*, have also been proposed by other authors [36, 55, 89, 262].

4.3 Terrain Generalization

The more data are available, the better a terrain can be represented. Modern acquisition techniques provide huge datasets that permit to represent terrain accurately. However, high representation accuracy is paid for in terms of high costs for storage and processing. Generalization techniques provide means to bring such costs into manageable bounds, by trading-off representation costs, and accuracy.

An *approximate terrain model* is a model built by using a reduced set of data. The approximation error ε is measured with respect to a reference model built on the whole data set: a common choice is the maximum difference between the elevation at a data point, and its interpolated elevation in the model. The accuracy of an approximated representation with respect to the reference model - which has no relation with the accuracy of samples with respect to reality - can be defined as the inverse of the error, namely $\frac{1}{1+\varepsilon}$.

An application can optimize performance by adopting an approximate terrain model whose accuracy is within a required threshold. In some cases, the threshold may be variable through the surface domain. For instance: higher accuracy may be needed in the proximity of important features, for the purpose of terrain analysis; accuracy may be decrasing with distance from a given viewpoint, for the purpose of landscape visualization. However, it is easy to transform a generalization problem with variable threshold into a problem with constant threshold, by performing a simple rescale of the input data through the inverse of the threshold function [194].

The ideal aim of *terrain generalization* is to achieve an optimal ratio between accuracy and size of representation. As for line simplification, there are two different optimization criteria:

- minimizing the number of vertices of the model for a given accuracy;
- maximizing the accuracy for a given number of vertices.

For the first problem a negative result has been proven by Agarwal and Suri [1]: they consider a polyhedral terrain (for simplicity, a TIN), and show that the problem of approximating it at a given accuracy with another TIN having a minimum number of triangles and vertices at arbitrary positions is NP-hard. It follows that also the second problem is NP-hard. It was conjectured that such problems remain NP-complete even if vertices of the approximate terrain are constrained to lie at original data points.

Only for the first problem, there exist algorithms that can achieve a suboptimal solution in polynomial time, while guaranteeing some bounds on its size. A first algorithm proposed by Agarwal and Suri [1] can build an approximate solution in $O(n^8)$ time, having $O(k \log k)$ triangles, where k is the number of triangles in the optimal solution. More recently, a randomized algorithm was proposed by Agarwal and Desikan [6] that can achieve size $O(k^2 \log^2 k)$ in $O(n^{2+\varepsilon} + k^3 \log^2 k \ \log \frac{n}{k})$ expected time. Both algorithms build a *simplicial partition*, defined as a set of disjoint triangles whose union covers all point data, and such that each triangle is compatible with the required accuracy. Such a set of triangles is completed to form a TIN. Finding a simplicial partition involves sophisticated techniques whose implementation is hardly feasible in practice; however, a simplified implementation of the latter algorithm has given empirical results that are comparable to those obtained with other methods.

An algorithm that solves the first problem with a greedy technique has been proposed by Silva et al. [253]. The approximate TIN is built incrementally by iteratively cutting triangles (*ears* or *bites*) from hollow polygons that span the domain: at each step the algorithm considers a polygon, and bites from it a triangle of maximal area that is compatible with the required accuracy. It is conjectured in that also this algorithm might guarantee a bound on the size of the solution, but proving this fact is still an open issue. This algorithm works with simple data structures, and requires a small amount of memory.

Most other algorithms use iterative methods based on heuristics, which try to select a "good" subset of the input dataset as a set of vertices for the approximate model. Such methods can be used either for the first or for the second problem, depending on the test adopted to stop iteration. *Refinement* heuristics start from a grid whose vertices are a very small subset of input data. New data points are iteratively inserted as vertices of the model, until the required constraints are satisfied. *Simplification* heuristics work in the opposite direction, by iteratively discarding points from an initial model built over the whole data set. In this case, points are selected at each iteration in order to cause the least possible increase in the error. Some authors have also adopted techniques that extract morphological features from the initial dataset in order to obtain constraints for the approximate TIN [33, 217, 245, 257, 283] (see also Section 5.2).

Approximate RSGs can only be obtained by subsampling the grid, while TINs are much more flexible because of their irregular structure. Many possible approximate models at different accuracies, with different sizes, can be built from a given TIN. In the case of contour maps, there are limited possibilities to vary the resolution by removing lines, while line simplification permits to obtain a wide range of different resolutions, and data sizes.

Refinement methods. In [101], Fowler and Little propose a method that starts from an RSG and builds an approximate TIN in two steps. First, point features are selected using the method described in [217], and an initial TIN is built based on the Delaunay triangulation of such points. Then, the model is refined by iteratively adding the point corresponding to the maximum approximation error and updating the Delaunay triangulation. A similar algorithm (not involving point features) was also developed by Franklin, and an efficient implementation is available in the public domain [108].

A similar strategy was applied in [257] by using a different method for feature selection, and a data dependent triangulation: the local optimization procedure (edge swapping) applied after each point insertion tries to minimize the error of the current model. However, convergence to a global minimum is not guaranteed. Variants of the same strategy, modified for a number of data dependent triangulations were also applied in [231].

Simplification methods. Lee proposed a simplification technique based on the Delaunay triangulation [174]. A Delaunay TIN is built first over the whole dataset. Then, vertices of the TIN are iteratively dropped, based on a criterion symmetric to the one of Fowler and Little: at each iteration, the point whose deletion causes the least increase in the error is dropped, and the triangulation is updated consequently.

Another simplification algorithm based on the hierarchical triangulation scheme of Kirkpatrick [161] has been proposed by de Berg and Dobrindt [50] with the purpose of building a multiresolution model (see Section 4.4). In this case, a maximal set of independent vertices is dropped at each iteration, and holes are filled by a Delaunay triangluation. Variants of the above algorithms have been analyzed and compared by De Floriani et al. [63].

A number of other simplification methods have been proposed for the more general case of *free form surfaces*. For surveys on such methods, see [135, 226].

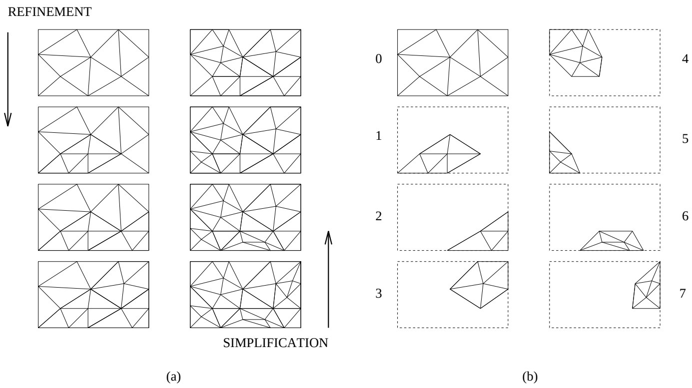

Figure 2: The sequence local modifications generated during the incremental refinement/simplification of a triangulation.

4.4 Multiresolution Terrain Models

Approximation algorithms reviewed in the previous section are usually time intensive: the construction of an approximate TIN from a dataset of 100K points can take minutes, or even hours, on state-of-the-art workstations, depending on the algorithm used.

On the other hand, many applications need to access and process terrain models at different accuracies in a real-time context. For example, in a flight simulator, a terrain must be rendered at a high detail only close to the observer. Since the viewpoint is continuously moving, the required accuracy is changing from frame to frame, while the system cannot wait until a suitable terrain approximation is recomputed each time from scratch. The ability to provide a representation whose accuracy is variable over the domain is often called *selective refinement*.

A so-called *LOD (level-of-detail) model* is a simple sequence of approximate representations at increasing levels of detail. LOD models are standard technology in graphics languages and packages, such as OpenInventor™ [209, 288], and VRML [277], and are used to improve efficiency of the rendering on the basis of distance from the observer. However, such models cannot support selective refinement.

In order to overcome such limitations, more sophisticated *multiresolution terrain models* have been developed in the literature. A multiresolution model must:

- perform selective refinement for any given accuracy in short (real) time;
- provide always conforming meshes, i.e., avoid cracks due to abrupt transition between different levels of detail within a mesh;
- have a size not much higher than the size of the model at full resolution.

In the following, we describe a general framework, called a *Multiresolution Mesh (MM)* [62, 224]. Multiresolution models proposed in the literature will be reviewed next as instances of such a framework.

Existing multiresolution models based on domain decomposition are all obtained from an initial model, which is progressively modified by a generalization algorithm (either by refinement or simplification). In a straightforward approach, a modification simply replaces a whole terrain representation with another one. More general modifications are *local*, i.e., they affect only a small part of the surface. Figure 2 shows a sequence of triangulated grids generated through incremental refinement/simplification and the sequence of modifications corresponding to the refinement process (including the initial grid). Two successive modifications can be either *independent*, if they affect disjoint parts of the mesh, or the second modification may *depend* on the first one, if some of the grid elements introduced by the first one are

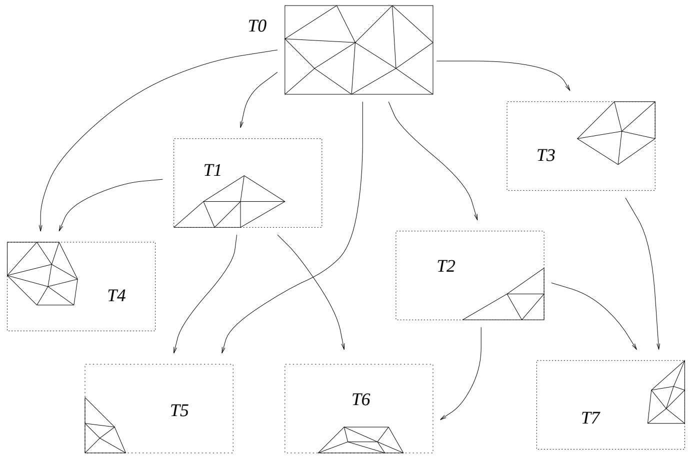

Figure 3: The MM corresponding to the sequence of local updates in Figure 2.

removed by the second one. Dependency means that the second modification cannot occur if the first one did not occur before. Each local modification can be represented by the set of mesh elements it introduces, which is called a *component*. Dependency relations define a partial order on the set of components, which can be represented by a DAG (see Figure 3).

The collection of such components, plus the partial order, form a Multiresolution Mesh (MM), which is at the heart of any multiresolution model. An MM has several interesting properties that have been investigated in detail in [224]. In particular, any consistent (i.e., dependency-closed) subset of components defines a mesh representing the terrain at a given level of detail: such a mesh is obtained by performing the corresponding local modifications in any temporal sequence consistent with the dependency relation. Moreover, it is also true that any possible mesh made of elements belonging to any component of the MM can be obtained from some consistent subset. Therefore, selective refinement on an MM consists of finding the consistent subset that is most suitable to a given LOD.

We generically refer to the capability of an MM to provide a larger or smaller number of consistent subsets (hence, different meshes) as its *expressive power*. The internal structure of any specific MM determines its expressive power. An MM with few large components with many mutual dependencies will have an expressive power much lower than an MM with many, small components with few mutual dependencies.

If an MM is obtained through refinement, any local modification increases resolution (and size) of the mesh locally. In this case the MM is said *increasing*. Even if an MM is built through simplification, it is possible to reverse modifications composing it, in order to obtain an increasing MM.

All multiresolution models proposed in the literature can be interpreted as increasing MMs: in practice, every model is characterized by special rules used for generating the local updates of the MM, by the data structures adopted, by the operations supported, and by the efficiency of accessing algorithms. However, vieweing the model as an MM helps understanding its expressive power independently of the specific data structures and algorithms.

Multiresolution models can be subdivided in two major classes, according to their structure:

- *tree-like models* are based on nested subdivisions; in order to give an interpretation of such models as MMs, the corresponding tree structure must be suitably translated into a DAG;
- *evolutionary models* directly store the evolution of a mesh through a refinement/simplification process; since such models are based on the concept of local modification, they are all straightforward specializations of the MM.

Tree-like Models. Early hierarchical models are based on the recursive subdivision of a domain either into four quadrants (*quadtree*) [242], or into four equilateral triangles (*quaternary triangulation*) [77, 118, 120]. The resulting hierarchy is represented by a quaternary tree. The main drawback with such models is a poor expressive power, due to the impossibility to combine elements from different levels of the tree into a conforming mesh. A surface corresponding to a non-conforming mesh is affected from cracks between elements from different levels. If a quadtree (or a quaternary triangulation) is regarded as a special instance of an MM, it is easy to see that all nodes at each level of the tree must be clustered to form a single component, hence giving a pyramid of full regular grids.

The basic problem with these models is that all edges of a region are split when the region is refined. In order to avoid cracks in the surface, the refinement of one region enforces refinement of its neighbors, and so on, propagating through the whole domain. Von Herzen and Barr [278] proposed a method to build conforming meshes from a quadtree that works in two steps: the extracted mesh is locally balanced first, by allowing adjacent elements to differ for no more than one level in the tree; next, each element is triangulated through a fixed pattern that depends on the levels of its neighbors.

Lately, other authors have worked on similar concepts: *hierarchies of right triangles* are based on the recursive split of a right triangle into two, by joining the midpoint of its longest edge to its opposite vertex [73, 90, 134, 178]. The hierarchy is described by a binary tree, and it can be interpreted as an MM by clustering triangles at the same level that are adjacent along their longest edges: components of the MM are either squares or diamonds, each formed by four adjacent triangles, except border components. Because the resulting MM is made of a high number of small components, it has a high expressive power. This model has been used successfully for rendering terrains in flight simulation through selective refinement [73, 90, 178].

All models based on regular subdivisions can be encoded by extremely compact and efficient data structures, by exploiting their algebraic properties (see [90, 134, 242] for details). On the other hand, such models can be used only if data points are distributed on a square grid.

Other tree-like models have been developed, based on irregular triangulations, which can work on arbitrary datasets, and can achieve a better ratio between size and accuracy of an extracted mesh, because no constraint is imposed on the vertex distribution [61, 246]. A triangle is refined by inserting a non-fixed number of points, either inside it, or on its edges, on the basis of an error-driven refinement criterion. Edges that survive across different levels of the hierarchy permit to combine surface patches from different levels of the tree.

Components of an MM corresponding to one such model are obtained by clustering nodes that refine adjacent triangles inserting vertices on their common edges. Each level of the tree is partitioned into a set of components, each bounded by edges that are not refined at that level. In general, a model with many refined edges will lead to an MM with few large componens, hence reducing its expressive power. On the other hand, a model with few refined edges may contain many slivery triangles: this gives unpleasant visual effects, and stability problems in numerical computation.

In [61], a comprehensive analysis of tree-like models is presented, covering several issues about data structures, neighbor finding, and selective refinement.

Evolutionary Models. Multiresolution is obtained by recording the evolution of a mesh through a sequence of local modifications (either refinement or simplification). Interpreting an evolutionary model as an MM is straightforward: each modification step performed by the algorithm generates a new component.

The forerunner of evolutionary models is the Delaunay pyramid proposed in [56]. Such a model is based on a sequence of Delaunay triangulations at increasing resolutions, obtained according to a predefined sequence of decreasing error thresholds. The transition from a triangulation to the next is obtained by iterative refinement [101]. The corresponding MM is obtained by decomposing such global modification into maximal local independent modifications. The expressive power of the model may be low because it does not trace the insertion of single vertices: the MM may have a relatively small number of large components.

The model proposed in [50] is also based on a sequence of Delaunay triangulations, but it is built bottom-up: each level is obtained from the previous one by eliminating an independent set of vertices of bounded degree. The corresponding MM is formed by a large number of components, namely one for each decimated vertex; moreover, the height of the DAG is logarithmic in the number of nodes. This property guarantees a good worst-case complexity of geometric search algorithms such as point location. The data structure is a direct implementation of the DAG. The algorithm for selective refinement proposed in [50] achieves output-sensitive optimal time, but it does not guarantee that the desired threshold will be fulfilled everywhere.

Evolutionary models based on a totally ordered sequences of local updates, rather than a DAG, can be encoded by very compact data structures. In order to perform selective refinement, the basic idea is to scan the list of updates, performing only those relevant to achieve the required accuracy. Unfortunately, a straight scan is not sufficient, since skipping an update may prevent performing some later update that is necessary to fulfill the accuracy. Therefore, dependencies between updates must be computed on-the-fly, thus involving complex, and computationally expensive algorithms.

An extremely compact model based on the Delaunay triangulation is proposed in [162]. In this case, a greedy refinement/decimation through on-line insertion/deletion of single vertices into a Delaunay triangulation is performed. The data structure stores only the simplest mesh, plus the sorted sequence of vertices inserted to refine it, each tagged with the approximation error of triangles incident at such vertex at time of its insertion. A mesh at uniform accuracy can be extracted through an on-line algorithm for Delaunay triangulation that inserts vertices from the sequence until the desired accuracy is obtained. An algorithm for selective refinement is also outlined, which perform complex tests based on the circumcircle criterion to find out vertex dependencies.

A similar model, called *Progressive Meshes (PM)*, is proposed in [145], based on a different update strategy. In this case, the basic local modification is *vertex split*, which expands a vertex to an edge. As for the previous model, the data structure stores only the simplest mesh, and the sequence of vertex splits. The extraction at uniform accuracy is faster than in [162] because no numerical computation is required. In the attempt to obtain efficient algorithms for selective refinement from a PM, more sophisticated data structures encoding also vertex dependencies have been proposed in [146, 186, 293]. In [293], a progressive mesh is built by collapsing independent sets of edges, following a technique similar to that used in [50].

The *Multi-Triangulation (MT)* is a straightforward implementation of the MM framework, where every component is a triangulation [62, 224]. An *explicit data structure* encodes all components of an MT as nodes, and maintains all arcs of the DAG describing the MT: every arc (T_i, T_j) is labeled by the set of triangles of T_i covered by T_j. Each leaf of the DAG is connected to an additional drain node, and each new arc is labeled with triangles of its source node that survive in the drain node. Based on this arc labeling, the explicit data structure represents triangles on arcs rather than on nodes of an MT.

This data structure supports the extraction of a mesh by collecting all triangles labeling arcs in a given *cut* of the DAG. An algorithm for selective refinement that finds the minimal triangulation for a given arbitrary threshold was described first in [224], and improved in [63]. The time complexity is linear in the size of the visited consistent subset. Independently, a similar algorithm was proposed in [26], achieving similar results.

A dynamic algorithm proposed recently [65] permits to update a mesh extracted from an MT after a small change in the threshold function. Such an algorithm only visits the portion of DAG separating the old and the new solution, respectively. This algorithm is especially efficient for generating gradually changing levels of detail in an interactive environment (e.g., flight simulation).

A cheaper, but less efficient *implicit data structure* for MTs built through refinement/simplification of a Delaunay triangulation has been proposed in [64]. In this case, each component consists of a fan of triangles incident at a vertex, and is represented in the structure by storing just the vertex. Implicit data structures can be defined for MT obtained as progressive meshes, and for the hierarchy of right triangles as well. Although the storage cost is reduced by these structures, empirical results show that the algorithm for selective refinement may run about ten times slower, and also the size of the output mesh may be about twice larger [64].

The *HyperTriangulation (HyT)* proposed in [40] can be also considered an implementation of an MT, but the data structure is completely different. Based on the observation that the boundary edges of a component T coincide with those of the triangulation covered by T, components in a hypertriangulation are interpreted in a three-dimensional space, where the third dimension corresponds to a *resolution axis*. Each component forms a "dome" in such a space, resting on the components it covers through its boundary edges. The data structure stores the triangles of the MT, and triangle adjacencies in the complex formed by such domes. Adjacency information supports algorithms based on domain traversal. Such an algorithm is used to perform selective refinement for threshold functions that are monotonically increasing with distance from a given viewpoint. The time complexity in the worst case is $O(n \log n)$ where n is the size of the HyT. Empirical results of the extraction algorithm show a performance of about 100K triangles per second on state-of-the-art workstations.

Methods based on wavelets. In the literature, also other kinds of multiresolution models have been proposed, which follow a *functional approach* rather than a geometric one. The basic idea is that a function can be decomposed into a simpler part at low resolution, together with a collection of perturbations, called *wavelet coefficients*, which define its details at progressively finer levels of resolution.

Wavelets have been widely used for multiresolution representation and compression of signals and images (see, e.g., [259, 260] for a survey), while their applications to surfaces is quite recent (see [181] for a survey, and [122] for an application to terrain surfaces). The discrete computation of wavelets requires a recursive subdivision of the domain into regular shapes, like equilateral triangles, or squares. Therefore, resulting hierarchies correspond to either quaternary triangulations, or quadtrees. The use of tools from numerical analysis is predominant, while computational geometry techniques do not play a relevant role in this context.

5 Terrain Analysis

Complex analyses of terrain models include [286]:

- computation of visibility maps, and solution of visibility-related problems;
- terrain generalization, and feature extraction;
- computation of watershed, and drainage networks;
- realistic visualization;
- path planning.

Terrain generalization issues have been treated in Section 4.3 for building approximated and multiresolution models. Related issues will also be considered in Section 5.2 on topographic features.

Rendering of DTMs and related information includes procedures for orthographic display (contours, hillshading, etc.), perspective display as well as advanced visualization techniques (interactive scene navigation, animation, etc.). Terrain visualization issues are related to viewshed computation problems (see Subsection 5.1). Also, some modeling issues related to effective real-time rendering of a DTM have been discussed in Section 4.4: approximate and variable resolution terrain models directly support compression for efficient rendering, and animation.

5.1 Visibility

Visibility problems are concerned with the computation of visibility information from a viewpoint which can lie outside or inside the domain, or with the use of visibility to solve optimization problems (see also Chapter 19). Examples of visibility computations are finding the horizon or computing the visible portions of the surface from a given viewpoint; in the latter case, the problem is a special case of the Hidden Surface Removal (HSR) for terrains. Visibility computation algorithms are the basis for solving optimization problems. Examples of optimization problems related to visibility are finding the minimum number of towers of a given height necessary to view an area of the terrain, or finding the minimum path with specified visibility characteristics (e.g., hidden paths, scenic paths). Applications include the location of fire towers, radar sites, radio, TV or telephone transmitters, path planning, navigation and orientation (see [203] for a survey).

Visibility computation problems can be classified into:

- *visibility queries*, which consist of determining whether a given entity located on the terrain is visible from a viewpoint, and possibly which portions of it are visible;
- *computation of visibility structures*, which provide information about the visibility of the terrain itself, i.e., which portions of the terrain are either visible, or invisible from a given viewpoint; knowing suitable visibility structures for a terrain also helps answering visibility queries.

A visibility query related to a point Q simply requires to determine whether Q is visible or not. For a non-point query object (a line, a region) one may ask for either a Boolean answer (e.g., "visible" means "at least partially visible"), or for a partition of the object into visible and non-visible portions.

The basic visibility structure for a terrain is the *viewshed* which is the collection of the surface portions visible from a viewpoint V. Another visibility structure is the *horizon* of a viewpoint V, which informally corresponds to the 'distal boundary' of the viewshed. Such reduced information can replace the viewshed in some applications, with the advantage of lower storage costs. More precisely, the horizon determines, the farthest point on the terrain that is visible from V, for every radial direction around V in the xy plane.

The result of visibility computations is affected by interpolation conventions adopted in the underlying terrain model. On TINs, a piecewise linear interpolation over each triangle is the standard choice. On RSGs, different GIS packages may use bilinear functions, or step functions, or other interpolation conventions; a common approach is considering a linear approximation of the edges, while disregarding the interior of the cells. Therefore, results can be different even in different implementations of the same algorithm [96].

The reminder of this subsection contains a survey of visibility algorithms. Algorithms working on TINs receive a special attention since they provide more interesting examples of applications of techniques from computational geometry.

Visibility query algorithms. The simplest visibility query problem consists in determining the mutual visibility of two points P and Q on a terrain. In a "brute-force" approach, this reduces to finding either the terrain edges (for a TIN), or grid cells (for an RSG) intersected by the vertical plane passing through segment PQ. For each intersected element (edge or cell) e, a test is performed to decide whether e lies above PQ, and the two points are reported as not visible in case of a positive answer for at least one of such tests.

A point visibility query can also be regarded as a special instance of the *ray shooting* problem on a polyhedral terrain. Given a polyhedral terrain $\mathcal{T}$, a viewpoint V and a view direction (θ, α), the *ray shooting* problem consists of determining the first face of $\mathcal{T}$ hit by a ray emanating from V with direction (θ, α). A method for answering repeated ray shooting queries from a fixed viewpoint on a terrain has been proposed by Cole and Sharir [45], with a logarithmic query time. They build a balanced binary tree, in which every node stores a partial horizon, computed for a subset of the edges of the terrain. A ray shooting query, represented by a view direction (θ, α), reduces to traversing a path on such tree, driven by a height comparison between the given ray and the horizon stored in the currently visited node. The data structure has size $O(n\alpha(n) \log n)$, where $\alpha(n)$ is the inverse of Ackermann's function (which is almost a constant).

Viewshed algorithms. On RSGs, the viewshed is usually represented in a discrete way, by marking each grid cell as visible or invisible [96]. Extended viewsheds are often considered, i.e., viewsheds enriched with additional information: for example, tagging each grid element with the number of viewpoints from which it is visible. Although a discrete viewshed is sometimes considered also for a TIN [175], viewshed computation in this case is usually performed by computing the visible portions of individual triangles. The continuos approach on RSGs is not used for two reasons: the size of the grid, and the complexity of geometric computations involving quadratic interpolating functions.

Visibility for an RSG cell can be computed easily by walking from the viewpoint to the given element, along the elements intersected by a line-of-sight, until either visibility is blocked, or the target is reached. This method performs redundant computations, since rays to different cells of the grid may overlap partially. van Kreveld [271] proposes a sweep-line approach to compute discrete viewsheds on an RSG in $O(n \log n)$ time on a $\sqrt{n} \times \sqrt{n}$ grid. Visibility computation on RSGs is conceptually simple, but it becomes computationally intensive due to the size of the grid.

Viewshed computation on a TIN is related to the Hidden Surface Removal (HSR) problem in a three-dimensional scene. The general quadratic upper bound to HSR for a polyhedral scene [247] applies also to the special case of a polyhedral terrain, thus giving a worst-case space complexity equal to $\Theta(n^2)$ for the viewshed of a TIN with n vertices.

The HSR problem for a three-dimensional scene has been extensively studied in the literature [49, 158, 191, 211, 247]. Some specific algorithms for polyhedral terrains have been developed [222, 229]. Another approach to HSR reduces to computing the upper envelope of a set of disjoint polygons in space [23, 69, 80]. The upper envelope of a set of polygons in 3D defines a partition of the xy plane into maximal connected regions, each of which is labeled with a polygon: the polygon π labeling a region R is the polygon with maximum height over R. The visible image of a polyhedral terrain with respect to a viewpoint V is equal to the upper envelope of the set of its faces, whose height is considered with respect to their distance from the viewplane. Computing the viewshed on a TIN reduces to computing the upper envelope of a set of semi-disjoint triangles.

The approaches used in existing algorithms for viewshed, HSR, or upper envelope computation can be classified as follows:

- an approach in which the faces are processed in *front-to-back order* from the viewpoint;
- a *divide-and-conquer* approach;
- a *sweep-line* approach;
- an *on-line* approach, which processes faces incrementally in any order;
- an approach based on special data structures for answering *ray shooting* queries;

Some algorithms also combine different approaches together. In the following, we provide a brief survey of existing algorithms, according to the above classification; the reviewed algorithms are summarized in table 2.

The *front-to-back approach* is the most popular method for performing HSR in a scene. A scene is *front-to-back sortable* with respect to a viewpoint if there exists an ordered sequence of its faces such that

Algorithm	Approach	Input	sortable
[80]	Divide-and-conquer	Triangles	No
[158]	Divide-and-conquer, front-to-back	Polygons	Yes
[222, 229]	Front-to-back	PTM	Yes
[208, 247]	Sweep-line	Polygons	No
[121]	Sweep-line	Polygons	Yes
[49]	Sweep-line, ray shooting	Polygons	No
[211]	Back-to-front, incremental	Triangles	Yes
[69]	On-line	Triangles	No
[49]	On-line, ray shooting	Polygons	No

Table 2: Major existing algorithms for region visibility computation.

no face coming later in the sequence can obscure a face preceding it. Not all scenes are sortable: terrains are always sortable for viewpoints lying outside the domain, while Delaunay-based TINs are guaranteed to be sortable also for internal viewpoints [57]. There are standard techniques for making a generic PTM sortable by splitting some of its faces.

Most front-to-back algorithms process faces in increasing distance order from the viewpoint, and maintain at each step the "contour" of the current visible image (i.e., the boundary of the union of the projections of the faces processed so far). When a new face is projected onto the viewplane, its edges are tested for intersections only against the edges of the contour. When the scene is a terrain, the contour is the same as the horizon, restricted to the subset of terrain faces which have been already examined. The algorithms by Reif and Sen [229], and Preparata and Vitter [222] are based on such approach, and run in $O((n+k)\log^2 n)$ time, where n is the size of the terrain model, and k is the size of the computed visible image.

The algorithm proposed by Overmars and Sharir in [211] represents a very effective method for computing the visible image of a sortable set of triangles. This algorithm combines the incremental and the divide-and-conquer approaches: triangles are added in back-to-front order, in groups, and at each step the visible image of the new set of triangles is merged with the old visible image. The algorithm works in $O(n\sqrt{c}\log n + k)$, time, where c is the maximum contour size; for a polyhedral terrain, $c = O(n\alpha(n))$, and thus we have an $O(n\sqrt{n\alpha(n)}\log n + k)$ complexity.

The *divide-and-conquer approach* includes the worst-case optimal algorithm by Edelsbrunner, Guibas and Sharir [80], which computes the upper envelope of a set of n disjoint triangles in optimal $O(n^2)$ time, and can be used for determining the viewshed on a TIN.

The method proposed by Katz, Overmars and Sharir [158] applies a divide-and-conquer strategy in a sortable three-dimensional scene to achieve an output-sensitive complexity. A balanced binary tree is built, whose leaves correspond to the faces (the left-to-right order on the leaves reflects the front-to-back order on the corresponding faces), and every internal node represents the union of the faces stored in its subtree. Then, the visible portion of the object associated with each node are computed during a traversal of the tree; at the end of the traversal, the visible portion of each face are found in the leaves of the tree. The time complexity is output-sensitive and, for a sortable PTM, it is equal to $O((n\alpha(n)+d)\log n)$, where n and d are, respectively, the input and the output size.

Sweep-line algorithms first project the whole scene onto the viewplane, and then traverse it by moving a vertical line from left to right. The time complexity is equal to $O((n+k)\log n)$, where k is the number of intersection points between terrain edges projected onto the viewplane [121, 208, 247]. The algorithm by de Berg [49] computes the viewshed on a TIN in an output-sensitive way, by combining a sweeping technique with the use of efficient data structure for answering *ray shooting* queries.

On-line incremental algorithms operate on a generic set of polygons or triangles, without any assumption about properties of the scene. An on-line randomized algorithm for computing the upper envelope of a set of triangles in the space (and thus the viewshed of a TIN) has been proposed by Boissonnat and Dobrindt [23]. The expected time complexity is equal to $O(n^2 \log n)$ for disjoint triangles.

Horizon algorithms. The horizon of a viewpoint on a PTM is equal to the *upper envelope* of the set of segments obtained by projecting the terrain edges onto a viewplane. The *upper envelope* of a set of segments in the plane (in our case, the viewplane) is a piecewise linear, possibly non-continuous line formed by the collection of points belonging to some of the segments and lying below no other segment in the given set.

The size of the upper envelope of p segments in the plane is $\Theta(p\alpha(p))$ [45]. Thus, the complexity of the horizon of a polyhedral terrain with n vertices is equal to $\Theta(n\alpha(n))$.

Computing the horizon of a viewpoint on a polyhedral terrain reduces to the computation of the upper envelope of a set of possibly intersecting segments in the plane. Upper envelope algorithms have been proposed based either on a divide-and-conquer or on an incremental paradigm. The classical divide-and-conquer scheme [10] runs in $O(p\alpha(p) \log p)$ time. A more sophisticated technique, which applies special care to split the set of segments into subsets with certain characteristics, achieves an optimal $O(p \log p)$ computational complexity [137]. A simple incremental approach has a complexity of $O(p^2\alpha(p))$, since adding the i-th segment can cause the modification of each of the $i\alpha(i)$ intervals of the current envelope, while a randomized incremental algorithm proposed in [60] has an expected time complexity of $O(p\alpha(p) \log p)$ for inserting all the given segments in random order.

Visibility computation on hierarchical models. Visibility computation problems have been addressed in the context of hierarchical terrain models by De Floriani and Magillo [66]. In this framework, two issues have been considered: the *computation* of visibility information, related to a terrain representation at a certain resolution level, and the *update* of such information, as the required resolution changes. Distance-based and sweep-line viewshed algorithms can work on a hierarchical terrain model by performing a traversal of its tree-like structure. Dynamic algorithms can be used to update viewsheds and horizons after local changes in the level of detail of a terrain representation (i.e., replacement of a subset of terrain faces and edges with a finer/coarser description) [69, 66].

Visibility-related problems. Examples of interesting application problems on a terrain, which can be solved based on visibility information, are:

- problems requiring the *placement of observation points* on a topographic surface according to certain requirements;
- *line-of-sight communication* problems;
- problems regarding the computation of *visibility paths* on a terrain, with certain properties.

Puppo and Marzano [225] investigate the connection between visibility-related problems in a discrete setting, and classical problems in graph theory: they show that graph algorithms can provide efficient and practical solutions to many visibility-related problems, belonging to all three classes listed above.

Viewpoint placement problems require placing several observation points on a terrain in such a way that each point of the terrain (or of a focus area on it) is visible from at least one observation point. Applications include the location of fire towers, artillery observers, and radar sites.

For a single observation point, algorithms running in polynomial time are known. If the height of the viewpoint is fixed, an existing solution can be determined in $O(n \log n)$ time, while the point with the lowest elevation, from which the entire terrain is visible, can be determined in $O(n \log^2 n)$ time [252], on a polyhedral terrain model with n vertices. Recently Zhu [295] improved the complexity to $O(n \log^* n)$.

The more general problem of determining the smallest number of observation points, which collectively see the entire terrain is known as the *guard allocation* problem and it is usually addressed by constraining viewpoints to be placed on the vertices of a polyhedral terrain. The problem has an exponential complexity, and it is equivalent to set-covering. Several heuristic algorithms to minimize the number of viewpoints, with fixed or variable height, are discussed in [175]. In the same paper, also the inverse problems are considered, that is, finding the optimal locations of viewpoints to maximize the visible area, assuming that the number and, possibly, the height of viewpoints as fixed. Bose et al. [24] reduce the guard allocation problem to a graph coloring problem and provide efficient heuristics to place viewpoints located at the vertices or on the edges.

Line-of-sight communication problems consist of finding a visibility network connecting two or more sites, such that every two consecutive nodes of the network are mutually visible. Applications are in the location of microwave transmitters for telephone, FM radio, television and digital data networks. Usually, sites are restricted to be at terrain vertices.

Finding the minimum number of relay towers necessary for line-of-sight communication between two sites can be formulated as a shortest path search in a graph where nodes are terrain vertices and arcs represent pair of mutually visible vertices, called a *visibility graph*. The problem of constructing a line-of-sight network between several sites is addressed in [59] by reducing it to the computation of a minimun Steiner tree on the visibility graph; memory requirements are kept low by computing the arcs of the graph on-line.

Visibility paths can be defined on a terrain, with application-dependent visibility characteristics. A *smuggler's path* is the shortest path, connecting two given points, such that no point on the path is visible from a predefined set of viewpoints. Conversely, a path where every point can be seen from all viewpoints is known as a *scenic path* [203]. Simple solutions can be obtained on a TIN by restricting the viewpoints to be vertices, and the path to pass along edges. The solution (if there exists) can be determined by first computing the vertices which are visible/invisible from all the viewpoints, and then applying a standard shortest path algorithm to the edges connecting them.

5.2 Topographic Features

Topographic features are special points, lines and regions that have a specific meaning for describing the shape of the terrain: they correspond to local differential properties of the terrain surface.

A point of the surface belongs to some characteristic class depending on the structure of the surface in its neighborhood. Point features are *peaks*, *pits* and *passes*. A peak is a point of relative maximum, a pit is a point of relative minimum, a pass is a saddle point for the surface, i.e., a point where there is a local maximum in one direction and a local minimum in a different direction.

Lineal features are *ridges* and *valleys*. A ridge is a curve consisting of ridge points: a point lies on a ridge if its neighborhood can be subdivided by a line passing through it, and such that the surface in each half-neighborhood is monotonically decreasing when moving away from the line. A ridge occurs where there is a local maximum in the surface in one direction; a valley occurs where there is a local minimum in one direction. Based on point and lineal features, Wolf [289] proposes the use of weighted surface networks for terrain description. A weighted surface network is a graph whose vertices and edges are point and lineal features, respectively; each edge is weighted with the height difference between its two endpoints. Wolf also suggests a method for contracting such networks, thus providing also a tool for terrain generalization.

Areal features are maximal regions on the terrain where the surface is *flat*, *convex* or *concave*. Such features are related to the curvature of the surface along two independent directions. For smooth surfaces they depend on the sign of the second derivative.

According to this classification, any point on a terrain can be labeled to belong to one of the above features by applying differential geometry tools. An interesting classification of topographic features can be found in [283].

In the discrete case, algorithms are essentially based on the analysis of the neighborhood of the points in the domain. In the literature, most methods have been developed for RSGs with the stepped model, because the neighborhood of each pixel is directly implied from the grid structure. The interpolation is commonly assumed to be constant on each grid cell, that is actually considered as a pixel in an image. The classical algorithm by Peucker and Douglas [217] classifies each pixel based on the heigth difference from its eigth neighbors in the grid. In [254] the method is improved to find also ridges and valleys wider than one pixel. Methods developed in the image processing literature can also be used [132, 163, 169]. Falcidieno and Spagnuolo[91] consider a virtual triangulation of the grid, where each cell is subdivided into four right triangles, and apply a method suitable for TINs (see below). Mc Cormack et al. [190] identify depressions and plateaus by propagating "seeds" consisting of connected regions of pixels with the same elevation, and bounded by pixels with a higher and lower elevation, respectively. Watson et al. [283], and Haralick [132] use RSGs with surface interpolation techniques.

In the case of TINs, point and lineal features are vertices and chains of edges in the underlying triangulation, respectively. Areal features are connected collections of triangles defining convex, concave or flat regions. A characterization of features on a TIN can be found in [103]. An algorithm for surface characterisation from a TIN, based on the local analysis of the dihedral angle formed by adjacent triangular patches, is proposed in [91]. Each TIN edge is labeled as convex, concave or flat depending on the slopes of its two incident triangles. Then, a triangle can be labeled based on the types of its edges (e.g., a triangle is convex if its three edges are concex), and convex, concave and flat surface regions are computed by aggregating triangles labeled in the same way. A compact representation of a TIN obtained by merging triangles into regions of uniform curvature is proposed as well.

5.3 Drainage Networks

Terrain drainage characteristics provide information on water resources, flood areas, erosion, and natural resource management. Manual quantification of terrain drainage characteristics is a tedious and time-consuming task. Thus, efforts have been undertaken to compute such characteristics from a terrain model.

On a terrain water flows downstream according to gravity and collects into streams, which join and form rivers. The collection of all streams and rivers forms the *drainage network* of the terrain. A drainage network can be seen as a forest in which the arcs are directed towards the pits. Hence, the computation of a drainage network is strongly related to the computation of point and lineal features of the underlying terrain (see [294] for a survey).

If we consider a terrain as a continuous surface, a drainage network can be characterized in terms of differential geometry (see [163, 164]). There is no accepted definition of the drainage network on an RSG. As noted by Yu et al. [294], the network is defined as the result of an algorithmic process. As for the case of feature extraction, local methods based on image processing techniques are applied on RSGs [72, 276].

Mark [188] proposes a global approach: all the cells are sorted in order of decreasing elevation and, initially, each cell is assigned one unit of fluid. Each cell in order adds its fluid to the lowest of its eight neighbors. At the end, cells receiving more than a certain quantity of fluid are declared to be on the drainage network. In this way, also the accumulation of water flowing in the terrain is modeled. Quantifying the accumulation of water is also useful for identifying drainage basins, i.e., surface regions formed by pixels whose water flow reaches the same pit.

The general assumption is that water flows from higher to lower heights. Flat regions and plateaus cause disturbs in water flow computation because the direction of flow is not well defined. Similar problems occur at depressions: in a real terrain, water cannot accumulate infinitely, thus it is necessary to determine the overflow points and directions for a depression. Many algorithms use simple heuristics based on local modification of the flow [150, 200, 143].

Mc Cormack et al. [190] note that the local modification of a drainage network without knowledge of its overall structure may have unpredictable effects. Therefore, they propose a feature-based approach to computation of drainage networks and basins on an RGS, which identifies inflow and outflow points for each depression and plateau, and defines the flow through the flat region as a line connecting them.

On a TIN, a definition of a drainage network, as a collection of all valley edges has been proposed by Frank et al. [103]. This definition has the drawback of possibly producing interrupted streams, because the actual location of a stream may not coincide with a triangle edge in the TIN. In addition, there is no concept of flow, so basins cannot be modeled. Yu et al. [294] introduce a broader definition of drainage network on a TIN, which also allows streams to pass through the interior of a triangle. Their definition is based on the following four assumptions: (i) at any point, water follows the steepest descent; (ii) watercourses can merge (thus the drainage network is a forest); (iii) watercourses end only at local minima; (iv) at any point there is a unique direction of steepest descent. The drainage network is defined as the collection of those points on the terrain that receive water from a region whose area exceeds a certain threshold value. They propose an algorithm for computing the network which runs in $O(n + k)$ time, where n is the size of the TIN and k the size of the drainage network. In [52] it has been proved that the complexity of the drainage network, as defined in [294], in a TIN with n vertices may be $O(n^3)$. By contrast, the drainage network defined according to Frank et al. [103] has a linear space complexity.

5.4 Path Problems

In this subsection, we deal with path problems on a terrain model (for path computation on a map see Subsection 3.5). Refer also to Chapter 15. Different optimality criteria for paths can be defined on a terrain [268]: measures of the quality of a path include the Euclidean length, the maximal height along the path, the total height difference, and the maximal slope of the path.

Several results appeared in the computational geometry literature on the shortest geodesic path problem. In particular algorithms have been proposed for computing such path on convex [201, 251] and on non-convex [34, 195] polyhedra. The best result is due to Chen and Han [34] who propose an $O(n^2)$ algorithm. Since polyhedral terrains are approximations of real surfaces, simpler and more practical algorithm which find an approximation to the optimal solution are sufficient and more desirable. Approximate algorithms for convex polyhedra are described in [3, 35, 133, 170], and for general polyhedra in [5].

Lanthier et al. [170] consider the problem of computing an approximate shortest path between two points on a TIN when each face has an associated weight. This is an instance of the weighted region problem discussed in Subsection 3.5. They propose practical algorithms based on the placement of Steiner points at edges of a polyhedral terrain, and on shortest paths on graphs. A good feature of such algorithms is that execution time and space requirements can be traded for accuracy, thus such algorithms can be used to build a sequence of shortest paths with increasing accuracy. Experimentally, the complexity of such an approach is $O(n \log n)$, while a worst case analysis leads to $O(n^5)$ and $O(n^3 \log n)$ complexities.

A heuristic technique based on discarding the parts of the terrain where the shortest path cannot pass through has been proposed in [15]; this approach, however, applies to regular triangulations, like quaternary ones.

In [268] algorithms are presented for the other quality measures discussed at the beginning of this subsection. In [54], algorithms are presented for answering path queries on a TIN in $O(\log n)$ time (with a preprocessing time of $O(n \log n)$). Such algorithmns support queries like deciding whether there exists a path between two points whose height decreases monotonically, or deciding whether there is a path joining two points and staying below a given height.

6 Three-dimensional GIS

Classical GISs deal with just two dimensions: terrains are not true three-dimensional objects, since most part of modelling and reasoning is done in their two-dimensional domain. Adding true three dimensionality to GIS is a more recent research area, in which computational geometry is directly involved.

Three-dimensional capabilities in a GIS system are required in cadastral and architectural applications for representation of buildings, and in geosciences and environmental modeling for representing the structure of either earth, or atmosphere (e.g., levels of rocks in geology; air flow, temperature and pressure in metereology). Three-dimensional geographic data can be handled with techniques inherited from 3D geometric modeling and CAD systems (see, e.g., the book by Mäntilä [187, 199]).

The shape of man-made objects (buildings, etc.) is usually regular and completely known, and directly modeled with classical CAD techniques. When representing physical phenomena (e.g., in geology), complex structures are known only at a finite set of sampled points, and interpolation must be used in order to recover a complete model of a three-dimensional area. The situation is similar to the case of terrain modeling, in one dimension higher. Jones [153] provides an overview of data models for three-dimensional objects in geology.

While a uniform and dense sampling of two-dimensional domains is easily achieved, three-dimensional sampling is usually sparse because of high costs. Thus, a regular three-dimensional grid would result in a useless waste of memory. Adaptive regular grids, such as octrees and their variants, are more appropriate. The octree [242] is a three-dimensional generalization of the quadtree. It is based on a recursive partition of a cubic universe into nested cubes of different sizes, which can be unfolded as a tree with degree eight. Each cubic cell is assigned an attribute value (e.g., a type of rock, a pressure value, etc.). Drawbacks of this representation are the necessity of a regular distribution of data points, and the rough approximation of boundaries between zones characterized by different attribute values: this is not admissible when information about the surface separating two different attributes (e.g., the surface between two different types of soil), must be maintained in the 3D model.

Irregular *tetrahedralizations* (i.e., the 3D extensions of triangulations) are also used. They provide an object-centered decomposition and adapt to the distribution of the sampled data. In particular, the Delaunay tetrahedralization extends the empty circle property of the Delaunay triangulation to an empty sphere property. A tetrahedral complex has the advantage of being able to include surface constraints (e.g., the contact surface between two layer of rock), as specific faces, edges, or vertices of the complex.

Efficient algorithms to build Delaunay tetrahedralizations of a point set in 3D are available in the computations geometry literature. Such algorithms can be classified into indirect algorithms (which compute the Delaunay tetrahedralization of a given set of points in 3D space by projecting the convex hull of a transformed set of points in 4D), incremental algorithms, based on the progressive insertion of points into an initial tetrahedralization [282, 151, 81], and algorithms which start from a single tetrahedron and progressively add new tetrahedra adjacent to the existing ones [12]. The problem of computing a constrained Delaunay tetrahedralization has been shown to be NP-complete [236]. Heuristic algorithms have been proposed which compute a conforming Delaunay tetrahedralization: the idea is replacing each data segment with a set of points, in such a way that the original segment will be a chain of edges in the output tetrahedralization [21]. Although there is no bound to the number of points which must be added to guarantee that all given segments are included in the resulting tetrahedralization, these methods seem good in practice. For details refer also to Chapter 5.

The Delaunay tetrahedralization has also been applied to reconstructing the boundary surface of solid objects. Input data for this problem are scattered points and/or segments lying on the object boundary. A common approach starts with the Delaunay tetrahedralization of the whole data, whose domain is the convex hull of the data set, and progressively removes tetrahedra in order to move internal points, or internal segments, on the boundary. The *sculpturing method* [20, 21] deletes tetrahedra iteratively based on heuristic considerations and topological constraints. The alpha-shape approach [84] delets all tetrahedra which can be "erased" by a sphere of a given radius, under the constraint that the sphere cannot pass through the data point. The radius is chosen through a binary search as the smallest value able to guarantee the consistency of the resulting shape. The alpha-shape method cannot resolve deep and narrow holes in the surface: such problem can be solved by applying the sculpturing method in a final step [14]. The method proposed by Veltkamp [267] for scattered segments is based on a different geometric structure, called the *γ-neighborhood graph*. The γ-neighborhood graph is a parametric graph that provides a continuous spectrum of graphs ranging from the empty to the complete graph (and including the Delaunay tetrahedralization as well). A suitable value of the parameter is chosen in order to obtain a feasible approximation of the object shape. A completely different approach is used in [144]: based on the available points, a function in 3D space is defined which pointwise extimates the distance from the unknown surface; then, a conventional algorithm (e.g., marching cubes) is used to extract the isosurface corresponding to the zero value of the function, which provides the final surface approximation.

7 Concluding Remarks

Computational geometry offers rigorous and powerful tools for solving a variety of geometric problems in GIS. In this survey, we have reviewed issues concerning efficient data structures to store geometric information on geographic data, and algorithms for basic spatial queries, overlay, map generalization, labeling, construction and conversion of terrain models, terrain approximation and multiresolution, visibility, extraction of topographic features, drainage networks, paths, as well as a handful of other specific problems.

At the state of the art, computational geometry tools are effective for many such problems, yet GIS practitioners often adopt heuristics that are less rigorous and efficient. This gap between theory and practice used to be justified by the difficulty of coding computational geometry algorithms, and especially because practical performance often did not follow from theoretical efficiency. Indeed, the search for asymptotic efficiency in the worst case (which hardly occurs) sometimes leads to very involved algorithms that perform poorly on real data configurations.

Although in some cases this is still true, recent advances, and research trends in computational geometry show that many theoretically rigorous tools either are, or will become soon, competitive also in practical applications [261]. The following issues have great relevance in making computational geometry effective, and easily available to practitioners.

- *Randomized algorithms* use more realistic models for evaluating the computational complexity, and may provide solutions as easy to program as brute-force approaches, and with provably good performance on real-world data. This class of algorithms is consolidated in computational geometry, and textbooks are already available [202]. Refer also to Chapter 16.
- *Robustness* in numerical computation is of primary importance, since errors due to floating-point arithmetic can compromise not only the accuracy, but even the correctness of results [142]. The problem of robustness has been faced by developing either methods for *exact arithmetic* [100], or algorithms that guarantee a correct result with *finite precision* [126], or *adaptive precision* [179] in geometric computations.
- *Computational geometry libraries* provide standard implementations of geometric algorithms, together with application programming interfaces that make such tools easily usable by practitioners. Developing and engineering libraries is perhaps the most successful way to export computational geometry technologies to applied fields. For instance, the LEDA library [204] provides a carefully engineered platform for developing discrete and geometric algorithms that is already used in industrial environments; the CGAL library [212] is under development, which can be integrated with LEDA, and offers tools for geometric robustness and specific algorithms for computational geometry; another geometric library under development is GeomLib [2].

From the point of view of researchers in computational geometry, the field of GIS is a highly stimulating arena, and a plentiful source of open problems. Strategic directions reported by the Computational Geometry Working Group [261] clearly indicate a list of hot issues - such as robustness, fine-grain complexity analysis, implementation, experimental evaluation, external-memory algorithms, dynamic and real-time computing, approximation algorithms, and parallel and distributed computing - all of which are relevant, and in some cases vital to GIS applications. The following open issues in GIS offer excellent cues for research.

- Realistic map models should be based on decompositions more general than planar subdivisions or CW complexes, in order to capture situations like regions with holes, and point alnd lineal features. Also, efficient data structures to encode such maps should be developed.
- Realistic queries in spatial databases always involve both geometric, and non-geometric aspects. A solution for such queries necessarily involves both geometric algorithms, and information retrieval techniques. A separate treatment of geometric and non-geometric issues often leads to inefficient solutions. An integrated approach to hybrid queries is currently lacking, and not even models for the evaluation of performances in resolving hybrid queries have been proposed yet.
- The management of large spatial datasets in external memory is also very important, and related to the previous issue. Most algorithms and data structures in computational geometry have been studied and developed for the primary storage, while this assumption is hardly realistic for any problem in GIS.
- Map generalization is an important task that involves many challenging geometric problems, and also requires the integration of geometric and non-geometric information. A comprehensive approach to generalization is still lacking. Serious achievements towards the automation of such task will probably require managing knowledge at a level much beyond the scope of geometry. However, a high benefit would come from an integrated treatment at least of all geometric problems.
- Compression and progressive trasmission of data is relevant, especially in the context of web-aware applications. While for terrains different techniques already exist, either based on geometric compression, or on wavelets, the compression and progressive transmission of maps is a more complex issue, since it involves both geometrical and toplological approximation.
- Multiresolution is currently a topic of great interest. Most of the work in multiresolution modeling of terrains (and, more generally, of surfaces) has been dealing with the problem of selective refinement, while most of the problems related with query processing, and terrain analysis at variable resolution are still an open issue. Facing such problems with a multiresolution approach may be vital to cope with the huge (practically worldwide) terrain databases that are becoming available.
- In the field of map processing, multiresolution is at an even earlier stage. As we discussed in Section 2.5, there are only a few empirical approaches based on heuristics, and few theoretically oriented works, based either on graph manipulation, or on transformations of cell complexes. The gap between theory and practice is still quite large. Important open problems are related to the design and development of efficient data structures, construction of multiple representations, and algorithms for answering spatial queries in a multiresolution framework, especially concerning spatial join (overlay), visibility, path computation, etc.
- Many optimization and approximation problems in GIS (e.g., terrain approximation, map labeling, some visibility-related problems) are intractable. Approximated algorithms that can run in reasonable time, and can guarantee bounds on the size of the result are very important to provide practical solutions to such problems.
- The field of 3D GIS, which is quite recent, offers many other cues of research. Much preliminary work is still needed in order to achieve a level of standardization sufficient at least to establish a common basis for researchers. In 3D, a classification of the different types of data, and of the relevant queries on them has still to be provided. Although both the kind of information and the queries

will be often strictly dependent on the specific application (e.g., modeling geological phenomena, or modeling buildings in an architectural environment), a serious analysis from practice to theory may lead to identifying general modeling and computational issues that offer a key to a more integrated approach to spatial information handling.

- Besides the issues listed above, there are plenty of specific problems that still need to be investigated, such as intersection-free line and map simplification, conflation and rubber sheeting of maps, edge matching and change detection, progressive transmission of maps and terrains, etc.

References

[1] Agarwal, P.K., Suri, S., Surface approximation and geometric partitions, *Proceedings 5th ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp.24-33.

[2] Agarwal, P.K., Goodrich, M.T., Kosaraju, S.R., Preparata, F.P., Tamassia, R., Vitter, J.S., Applicable and robust geometric computing, 1995, available from http://www.cs.brown.edu/cgc/.

[3] Agarwal, P.K., Har-Peled, S., Sharir, M., Varadarajan, K.R., Approximating shortest paths on a convex polytope in three dimensions, *Journal of the ACM*, 1997.

[4] Agarwal, P.K., van Kreveld, M., Suri, S., Label placement by maximum independent sets in rectangles, *Proceedings 9th Canadian Conference on Computational Geometry*, August 1997.

[5] Agarwal, P.K., Varadarajan, K.R., Approximating shortest paths on a polyhedron, *Proceedings 38th IEEE Symposium on Foundations of Computer Science*, 1997.

[6] Agarwal, P.K., Desikan, P.K., An efficient algorithm for terrain simplification, *Proceedings 8th ACM-SIAM Symposium on Discrete Algorithms*, 1997.

[7] Andrews, D.S., Snoeyink, J., Boritz, J., Chan, T., Denham, G., Harrison, J., Zhu, C., Further comparison of algorithms for geometric intersection problems, *Proceedings 6th International Symposium on Spatial Data Handling*, 1994, pp. 709-724.

[8] Aref, W.G., Samet, H., Uniquely reporting spatial objects: yet another operation for comparing spatial data structures, *Proceedings 5th International Symposium on Spatial Data Handling*, Charleston, SC, 1992, pp.178-189.

[9] Aronoff, S., *Geographic Information Systems: A Management Perspective*, WDL Publications, Ottawa, 1989.

[10] Atallah, M., Dynamic computational geometry, *Proceedings 24th IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, Baltimora, 1983, pp.92-99.

[11] Aurenhammer, F., Voronoi diagrams - A survey of fundamental geometric data structures, *ACM Computing Surveys*, 23, 3, 1991, pp.366-405.

[12] Avis, D., Battacharya, B.K., Algorithms for computing d-dimensional Voronoi diagrams and their duals, *Advances in Computing Research*, 1, JAI Press, F.P. Preparata (editor), 1983, pp.159-180.´

[13] Bajaj, C.L., Pascucci, V., Schikore, D.R., Fast isocontouring for improved interactivity, *Proceedings 1996 SPIE Symposium on Volume Visualization*, San Francisco, 1996, pp.39-46.

[14] Bajaj, C.L., Bernardini, F., Chen, J., Schikore, D.R., Automatic Reconstruction of 3D CAD Models, *Theory and Practice of Geometric Modeling*, W. Straßer, R. Klein, R. Rau (Editors), Springer-Verlag, 1996.

[15] Barrera, R., Vazques-Gomez, J., A shortest path algorithm for hierarchical terrain models, *Proceedings AUTO-CARTO 9*, 1989, pp.156-163.

[16] Baumgart, B., A polyhedral representation for computer vision, *Proceedings of the AFIPS National Computer Conference*, 1975, pp.589-596.

[17] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B., The R*-tree: an efficient and robust access method for points and rectangles, *Proceedings ACM SIGMOD*, 1990, pp.81-90.

[18] Bentley, J.L., Ottman, T.A., Algorithms for reporting and counting geometric intersections, *IEEE Transactions on Computers*, 28, 9, 1979, pp.643-647.

[19] Bertolotto, M., De Floriani, L., Puppo, E., Multiresolution topological maps, *Advanced Geographic Data Modelling - Spatial Data Modelling and Query Languages for 2D and 3D Applications*, M. Molenaar, S. De Hoop (Eds.), Publication on Geodesy - New Series, 40, 1994, pp.179-190.

[20] Boissonnat, J.D., Geometric structures for three-dimensional shape representation, *ACM Transactions on Graphics*, 3(4), 1984, pp. 266-286.

[21] Boissonnat, J.D., Faugeras, O.D., Le Bras-Mehlman, E., Representing Stereo Data with the Delaunay Triangulation, *Artificial Intelligence*, 44, 1990, pp.41-89.

[22] Boissonnat, J.D., Devillers, O., Schott, R., Tailland, M., Yvinec, M., Application of random sampling to on-line algorithms in computational geometry, *Discrete and Computational Geometry*, 8, 1992, pp.51-71.

[23] Boissonnat, J.D., and Dobrindt, K., On-line construction of the upper envelope of triangles in $\mathbb{R}^3$, *Proceedings 4th Canadian Conference on Computational Geometry*, C.A.Wang (Ed.), Memorial, Newfoundland, 1992, pp.311-315.

[24] Bose, P., Shermer, T., Toussaint, G. and Zhu, B., Guarding polyhedral terrains, *Computational Geometry - Theory and Applications*, 7, 3, 1997, pp.173-185.

[25] Brisson, E., Representing geometric structures in d dimensions: topology and order, *Discrete and Computational Geometry*, 9, 4, 1993, pp.387-426.

[26] Brown, P.J.C., A fast algorithm for selective refinement of terrain meshes, *Proceedings COMPUGRAPHICS 96*, GRASP, December 1996, pp.70-82.

[27] Burrough, P.A., *Principles of Geographic Information Systems for Land Resources Assessment*, Clarendon Press, Oxford, 1986.

[28] Carver, S.J., Brunson, C.F., Vector to raster conversion error and feature complexity: an empirical study using simulated data, *International Journal of Geographical Information Systems*, 8, 3, 1994, pp.261-270.

[29] Carrara, A., Bitelli, G., Carlà, R., Comparison of techniques for generating digital terrain models from contour lines, *International Journal of Geographical Information Science*, 11, 5, pp.451-474.

[30] Chazelle, B., Edelsbrunner, H., An optimal algorithm for intersecting line segment in the plane, *Journal of the ACM*, 39, 1, 1992, pp.1-54.

[31] Chazelle, B., Edelsbrunner, H., Guibas, L.J., Sharir, M., Algorithms for bichromatic line-segment problem and polyhedral terrains, *Algorithmica*, 11, Springer Verlag, New York, 1994, pp.116-132.

[32] Chan, T., A simple trapezoidal sweep algorithm for reporting red/blue segment intersections, *Proceedings 6th Canadian Conference on Computational Geometry*, 1994, pp.263-268.

[33] Chen, Z.-T., Guevara, J.A., Systematic selection of very important points (VIP) from digital terain model for constructing triangular irregular networks, *Proceedings AUTO-CARTO 8*, Baltimore, MD, USA, 1987, pp.50-56.

[34] Chen, J., Han, Y., Shortest paths on a polyhedron,, *Proceedings 6th ACM Symposium on Computational Geometry*, 1990, pp.360-369.

[35] Choi, J., Sellen, J., Yap, C.K., Approximate Euclidean Shortest Path in 3-Space, *Proceedings 10th ACM Symposium on Compututational Geometry*, 1994, pp.41-48.

[36] Chow, M.M., Optimized geometry compression for real-time rendering, *Proceedings IEEE Visualization '97*, 1997, pp.347-354.

[37] Christensen, J., Fitting a triangulation to contour lines, *Proceedings AUTOCARTO 8*, Baltimore, Maryland, 1987, pp.57-67.

[38] Christensen, J., Marks, J., Stuart, S., An empirical study of algorithms for point-feature placement, *ACM Transactions on Graphics*, 14, 3, 1995, pp.203-232.

[39] Ciampalini, A., Cignoni, P., Montani, C., Scopigno, R., Multiresolution decimation based on global error, *The Visual Computer*, 13(5), 1997, pp.228–246.

[40] Cignoni, P., Puppo, E., Scopigno, R., Representation and Visualization of Terrain Surfaces at Variable Resolution, *The Visual Computer*, 13, 1997, in press.

[41] Clarkson, K.L., Shor, P.W., Application of random sampling in computer geometry, *Discrete and Computational Geometry*, 4, 5, 1989, pp.387-421.

[42] Clementini, E., Di Felice, P., A Comparison of Methods for Representing Topological Relationships. *Information Sciences*, 3, 1995, pp.149-178.

[43] Clementini, E., Di Felice, P., A Model for Representing Topological Relationships Between Complex Geometric Features in Spatial Databases, *Information Sciences*, 90, 1-4, 1996, pp.121-136.

[44] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., Wright, W., Simplification envelopes, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '96)*, 1996, pp.119-128.

[45] Cole, R., and Sharir, M., 1986, Visibility problems for polyhedral terrains, *Journal of Symbolic Computation*, 17, 1989, pp.11-30.

[46] Corbett, J.P., Topological principles in cartography, *Technical Report* 48, U.S. Bureau of Census, USA., 1979.

[47] Cromley, R.G., A spatial allocation analysis of the point annotation problem, *Proceedings 2nd International Symposium on Spatial Data Handling*, Seattle, WA, July 1986, pp.38-49.

[48] D'Azavedo, E.F., Simpson, R.B., On optimal interpolation triangle incidences, *SIAM Journal on Scientific and Statistic Computing*, 20, 6, 1989, pp.1063-1075.

[49] de Berg, M., Ray shooting, Depth Orders and Hidden Surface Removal, *Lecture Notes in Computer Science*, 703, 1993, Springer Verlag, New York.

[50] de Berg, M., Dobrindt, K.T.G., On the levels of detail in terrains, *11th ACM Symposium on Computational Geometry*, Vancouver, BC (Canada), June 5-7, 1995.

[51] de Berg, M. van Kreveld, M., Shirra, S., 1995, A new approach to subdivision simplification, *Technical Papers, ACSM/ASPRS Annual Convention*, 4 (AUTO-CARTO 12), pp.79-88.

[52] de Berg, M., Bose, P., Dobrindt, K., van Kreveld, M., Overmars, M., de Groot, M., Roos, T., Snoeyink, J., Yu, S., The complexity of rivers in triangulated terrains, *Proceedings 8th Canadian Conference on Computational Geometry*, Ottawa, 1996, pp.325-330.

[53] de Berg, M., van Kreveld, M., van Oostrum, R., Overmars, M., Simple traversal of a subdivision without extra storage, *International Journal of Geographic Information Science*, 11, 4, June 1997, pp.359-373.

[54] de Berg, M., van Kreveld, M., Trekking in the alps without freezing or getting tired, *Algorithmica*, 18, pp.306-323.

[55] Deering, M., Geometry compression, *Computer Graphics (SIGGRAPH'95)*, 1995, pp.13-20.

[56] De Floriani, L., A pyramidal data structure for triangle-based surface description, *IEEE Computer Graphics and Applications*, 8, March 1989, pp.67-78.

[57] De Floriani, L., Falcidieno, B., Nagy, G., Pienovi, C., On sorting triangles in a Delaunay tesselation, *Algorithmica*, 6, Springer-Verlag, New York, 1991, pp.522-535.

[58] De Floriani, L., Marzano, P. Puppo, E., Spatial queries and data models, *Spatial Information Theory - A theoretical basis for GIS*, A.U. Frank, I. Campari (Eds.), *Lecture Notes in Computer Science*, 716 , Springer-Verlag, 1993, pp.113-138

[59] De Floriani, L., Marzano, P., Puppo, E., Line-of-sight communication on a terrain models, *International Journal of Geographic Information Systems*, 8, 4, 1994, Taylor & Francis, London, pp.329-342.

[60] De Floriani, L., Magillo, P., Horizon computation on a hierarchical terrain model, *The Visual Computer: An International Journal of Computer Graphics*, Springer Verlag, New York, 11, 1995, pp.134-149.

[61] De Floriani,L., Puppo, E., Hierarchical triangulation for multiresolution surface description, *ACM Transactions on Graphics*, 14, 4, 1995, pp.363-411.

[62] De Floriani, L., Magillo, P., Puppo, E., A formal approach to multiresolution modeling, *Geometric Modeling: Theory and Practice*, W. Straßer, R. Klein, R. Rau (Eds.), Springer-Verlag, 1996.

[63] De Floriani, L., Magillo, P., Puppo, E., Building and traversing a surface at variable resolution, *Proceedings IEEE Visualization 97*, Phoenix, AZ (USA), October 1997, pp. 103–110.

[64] De Floriani, L., Magillo, P., Puppo, E., Efficient Encoding and Retrieval of Triangle Meshes at Variable Resolution, *Technical Report* DISI-TR-97-01, Department of Computer and Information Sciences, University of Genova, Genova, Italy, 1997.

[65] De Floriani, L., Magillo, P., Puppo, E., Efficient implementation of Multi-Triangulations, Proceedings IEEE Visualization'98, Triangle Park, North Carolina (October 1998), pp.43–50.

[66] De Floriani, L., Magillo, P., Visibility Computations on Hierarchical Triangulated Terrain Models, *Geoinformatica*, 1 (3), 1997, pp. 219–250.

[67] Delis, V., Hadzilacos, T., On the assessment of generalization consistency, *Proceedings 5th Symposium on Spatial Databases* (SSD97), Berlin, July 1997.

[68] Dettori, G., Puppo, E., How generalization interacts with the topological and geometric structure of maps, *Proceedings 7th International Symposium on Spatial Data Handling*, Delft (The Netherlands), August 1996.

[69] Dobrindt, K., Yvinec, M., Remembering conflicts in history yields dynamic algorithms, *Proceedings 4th Annual International Symposium on Applied Computing* (ISAAC '93), *Lecture Notes in Computer Science*, 762, 1993, pp.21-30.

[70] Doerschler, J.S., Freeman, H., A rule-based system for dense-map name placement, *Communications of the ACM*, 35, 1, 1992, pp.68-79.

[71] Dougenik, J.A., Chrisman, N.R., Niemeyer, D.R., Ad algorithm to construct continuous area cartograms, *Professional Geographer*, 37, 1985, pp.75-81.

[72] Douglas, D.H., Peucker, T.K., Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *The Canadian Cartographer*, 10, 2, 1973, pp.112-122.

[73] Duchaineau, M., Wolinsky, M., Sigeti, D.E., Miller, M.C., Aldrich, C., Mineed-Weinstein, M.B., ROAMing terrain: Real-time Optimally Adapting Meshes, *Proceedings IEEE Visualization'97*, 1997, pp.81-88.

[74] Duda, R.O., Hart, P.E., *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973.

[75] Dutton, G., Geodesic modelling of planetary relief, *Cartographica*, 21, 1988, pp.188-207.

[76] Dutton, G., Polyhedral hierarchical tessellation: the shape of GIS to come. *Geo Info Systems*, 2(1), 1991, pp.49-55.

[77] Dutton, G., Improving locational specificity of map data - a multiresolution, metadata-driven approach and notation, *International Journal of Geographic Information Systems*, 10, 3, 1996, pp.253-268.

[78] Dyn, N., Levin, D., Rippa, S., Data dependent triangulations for piecewise linear interpolation, *IMA Journal of Numerical Analysis*, 10, 1990, pp.137-154.

[79] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer Verlag, 1987.

[80] Edelsbrunner, H., Guibas, L.J., Sharir, M., The upper envelope of piecewise linear functions: algorithms and applications, *Discrete and Computational Geometry*, 4,1989, pp.311-336.

[81] Edelsbrunner, H., Shah, N.R., Incremental topological flipping works for regular triangulations, *Proceedings 8th Annual ACM Symposium on Computational Geometry*, 1992, pp.43-52.

[82] Edelsbrunner, H., Waupotitsch, R., A combinatorial approach to cartograms, *Computational Geometry Theory and Applications*, 7, 1997, pp.343-360.

[83] Edelsbrunner, H., Tan, T.S., An Upper Bound for Conforming Delaunay Triangulations, *Discrete Computational Geometry*, 10, 1993, pp.197-213.

[84] Edelsbrunner, H., Mucke, E.P., Three-Dimensional Alpha shapes, *ACM Transaction on Graphics*, 13, 1, 1994, pp.43-72.

[85] Egenhofer, M., Franzosa, R., Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5, 2, 1991, pp.161-174.

[86] Egenhofer, M., Clementini, E., Di Felice, P., Topological relations between regions with holes, *International Journal of Geographical Information Systems*, 8, 2, 1994, pp.129-142

[87] Egenhofer, M., Clementini, E., Di Felice, P., Evaluating inconsistencies among multiple representations, *Proceedings 6th International Symposium on Spatial Data Handling*, Edinburgh, Scotland, 1994, pp.901-920

[88] Eklundh, L., and Mårtensson, U., Rapide generation of digital elevation models from topographic maps, *International Journal of Geographical Information Systems*, 9, 3, 1995, pp.329-340.

[89] Evans, F., Skiena, S., Varshney, A., Optimizing triangle strips for fast rendering, *Proceedings IEEE VIsualization '96*, 1996, pp. 319-326.

[90] Evans, W., Kirkpatrick, D., Townsend, G., Right Triangular Irregular Networks, *Technical Report* 97-09, University of Arizona, 1997.

[91] Falcidieno, B., Spagnuolo, M., A new method for the characterization of topographic surfaces, *International Journal of Geographical Information Systems*, 5, 4, 1991, pp.397-412.

[92] Faloutsos, C., Kamel, I., Hilbert R-tree: An improved R-tree using fractals, *Proceedings International Conference on Very Large Data Bases* (VLDB), 1994.

[93] Faloutsos, C., Kamel, I., Beyond Uniformity and Independence: Analysis of R-trees using the concept of fractal dimension, *Proceedings ACM SIGACT-SIGMOD-SIGART PODS*, 1994, pp.4-13.

[94] Finke, U., Hinrichs, K.H., The quad view data structure - A representation for planar subdivisions, *Advances in Spatial Databases*, M.J. Egenhofer, J.R. Herring, Editors, *Lecture Notes in Computer Science*, 951, Springer, 1995, pp.29-46.

[95] Finke, U., Hinrichs, K.H., Overlaying simply connected planar subdivisions in linear time, *Proccedings 11th Canadian Conference on Computational Geometry*, 1995, pp.119-126.

[96] Fisher, P.F., Algorithm and implementation uncertainty in viewshed analysis, *International Journal of Geographical Information Systems*, 7, 4, 1993, pp. 331-347.

[97] Flajolet, P., Gonnet, G.H., Puech, C., Robson, J.M., Analytic variations on quedtrees, *Algorithmica*, 10, 6, 1993, pp.473-500.

[98] Foley, D.J., Van Dam, A., Feiner, S.K., and Hughes, J.F., 1990, *Computer Graphics: Principles and Practice*, Reading, MA, Addison Wesley.

[99] Formann, M., Wagner, F., A packing problem with applications to lettering of maps, *Proceedings 7th ACM Symposium on Computational Geometry*, 1991, pp.281-288.

[100] Fortune, S., Van Wyk, C., Efficient exact arithmetic for computational geometry, *Proceedings 9th ACM Symposium on Computational Geometry*, 1993, pp.163-172.

[101] Fowler, R.F., Little, J.J., Automatic extraction of irregular digital terrain models, *Computer Graphics*, 13, 1979, pp.199-207.

[102] Frank, A.U., Kuhn, W., Cell graph: a provable correct method for the storage of geometry, *Proceedings 2nd Symposium on Spatial Data Handling* (SDH'86), Seattle, WA, 1986, pp.411-436.

[103] Frank, A.U., Palmer, B., Robinson, V.B., Formal methods for the accurate definition of some fundamental terms in physical geography, *Proceedings 2nd International Symposium on Spatial Data Handling*, 1986, pp.585-599.

[104] Franklin, W.R., Efficient intersection calculations in large databases, *Proceedings International Cartographic Association 14th World Conference*, Budapest, 1989, pp.A62-A63.

[105] Franklin, W.R., Kankanhalli, M., Narayanaswami, C., Geometric computing and the uniform grid data technique, *Computer Aided Design*, 21, 7, 1989, pp.410-420.

[106] Frank, A.U., Spatial concepts, geometric data models, and data structures, *Computers and Geosciences*, 18, 4, 1992, pp.409-417.

[107] Frank, A., Timpf, S., Multiple representations for cartographic objects in a multiscale tree – An intelligent graphical zoom. *Computers & Graphics*, 18, 6, 1994, pp.823-829

[108] Franklin, W.R., Triangulated irregular network to approximate digital terrain, *Technical Report*, ECSE Dept., RPI, Troy (NY), 1994. Manuscript and code available on ftp://ftp.cs.rpi.edu/pub/franklin/

[109] Fuchs, H., Uselton, S.P., Zedem, Z., Optimal surface reconstruction from planar contours, *Communications of the ACM*, 20, 10, 1977, pp.693-702.

[110] Garcia, A.B., Niciera, C.G., Merè, J.B.O., Diax, A.M., A contour-line based triangulation algorithm, *Proceedings 5th International Symposium on Spatial Data Handling*, 1992, pp.411-421.

[111] Ganapathy, S., Dennehy, T.G., A new general triangulation method for planar contours, *Computer Graphics*, 16, 3, 1982, pp.69-75.

[112] Gewali, L., Meng, A., Mitchell, J.S.B., Ntafos, S., Path planning in $0/1/\infty$ weighted regions with applications, *Proceedings 4th Symposium on Computational Geometry*, 1988, pp.266-278.

[113] Ghosh, S.K., Computing the visibility polygon from a convex set and related problems, *Journal of Algorithms*, 12, 1991, pp.75-95.

[114] Gold, C., Problems with handling spatial data - the Voronoi approach, *CISM Journal*, 45, 1991, pp.65-80.

[115] Gold, C.M. , An object-based dynamic spatial model, and its application in the development of a user-friendly digitizing system, *Proceedings 5th International Symposium on Spatial Data Handling*, Charleston, IGU Commission on GIS, 1992, pp.495-504.

[116] Gold, C., Dynamic data structures, *Advanced Geographic Data Modelling - Spatial Data Modelling and Query Languages for 2D and 3D Applications*, M. Molenaar, S. De Hoop (Eds.), Netherlands Geodetic Commission, Publication on Geodesy - New Series, 40, 1994, pp.121-128.

[117] Gold, C., Roos, T., Surface modeling with guaranteed consistency - an object-based approach, *Proceedings IGIS'94: Geographic Information Systems*, Nievergelt, J., Roos, T., Schack, H.J., Widmayer, P. (Eds.), *Lecture Notes in Computer Science*, 884, Springer-Verlag, 1994, pp.70-85.

[118] Gomez, D., Guzman, A., A digital model for three-dimensional surface representation, *Geoprocessing*, 1, 1979, pp.53-70.

[119] Goodchild, M.F., Kemp, K.K., Poiker, T.K., *NCGIA Core Curriculum*, National Center for Geographic Information Analysis, University of California, Santa Barbara, USA, 1989, also available at http://www.env.gov.bc.ca/ tclover/giscourse/ToC.html.

[120] Goodchild M.F., Shiren Y., A hierarchical data structure for global geographic information systems, *Computer Vision, Graphics and Image Processing*, 54, 1992, pp.31-44.

[121] Goodrich, M.T., A polygonal approach to hidden line and hidden surface elimination, *Graphical Models and Image Processing*, 54, 1, 1992, pp.1-12.

[122] Gross, M.H., Staadt, O.G., Gatti, R., Efficient triangular surface approximation using wavelets and quadtree data structures, *IEEE Transactions on Visualization and Computer Graphics*, 2(2), pp.130-143.

[123] Guibas, L.J., Stolfi, J., Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Transactions on Graphics*, 4, 1985, pp.74-123.

[124] Guibas, L.J., Seidel, R., Computing convolutions by reciprocal search, *Discrete and Computational Geometry*, 2, 1987, pp.175-193.

[125] Guibas, L.J., Hershberger, J.E., Mitchell, J.S.B., Snoeyink, J.S., Approximating polygons and subdivisions with minimum-link paths, *International Journal of Computational Geometry and Applications*, 3, 4, 1983, pp.383-415.

[126] Guibas, L.J., Marimont, D.H., Rounding arrangements dynamically, *Proceedings 11th ACM Symposium on Computational Geometry*, 1995, pp.190-199.

[127] Guttman, A., R-trees: a dynamic index structure for spatial searching, *Proceedings ACM SIGMOD*, 1984, pp.47-57.

[128] Güting, R.H., Schneider, M., Realms: a foundation for spatial data types in database systems, *Proceedings 3rd International Symposium on Large Spatial Databases*, Singapore, 1993, pp.14-35.

[129] Güting, R.H., Ridder, T., Schneider, M., Implementation of the ROSE algebra: efficient algorithms for realm-based spatial data types, *Proceedings 4th International Symposium on Large Spatial Databases*, Portland, ME, Lecture Notes in Computer Science, 951, Springer-Verlag,1995, pp.216-239.

[130] Hadzilacos, T., Tryfona, N., A model for expressing topological integrity constraints in geographic databases, *Lecture Notes in Computer Science*, 639, Springer-Verlag, 1992, pp.252-268.

[131] Hamman, B., A data reduction scheme for triangulated surfaces, *Computer Aided Geometric Design*, 11, 1994, pp.197-214.

[132] Haralick, R.M., Ridges and valleys on digital images, *Computer Vision, Graphics and Inage Processing*, 22, 1983, pp.28-38.

[133] Har-Peled, S., Sharir, M., Varadarajan, K.R., Approximate Shortest Paths on a Convex Polytope in Three Dimensions, *Proceedings 12th Annu. ACM Sympos. Comput. Geom.*, 1996, pp.329-338.

[134] Hebert, D.J., Kim, H.-J., Image encoding with triangulation wavelets, *Proceedings SPIE*, 2569, 1, 1995, pp.381-392.

[135] Heckbert, P., Garland, M., Survey of polygonal surface simplification algorithms, *Siggraph'97 Course Notes*, 1997.

[136] Herring, J., TIGRIS: topologically integrated GIS, *Proceedings AUTO-CARTO 8*, ASPRS/ACSM, Baltimore, MD, pp.282-291, March 1987.

[137] Hershberger, J., Finding the upper envelope of n line segments in $O(n \log n)$ time, *Information Processing Letters*, 33, 1989, pp.169-174.

[138] Hershberger, J., Snoeyink, J., Computing minimum length paths of a given homotopy class, *Computational Geometry - Theory and Applications*, 4, 1994, pp.63-97.

[139] Hershberger, J., Snoeyink, J., Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time, *Proceedings 5th Annual Workshop on Algorithms and Data Structures*, August 1997, Halifax, Nova Scotia, Canada.

[140] Hjaltason, G.R., Samet, H., Ranking in spatial databases, *Advances in Spatial Databases, Lecture Notes in Computer Science*, 951, Springer-Verlag, 1995, pp.83-95.

[141] Hoel, E.G., Samet, H., Efficient processing of spatial queries in line segment database, *Advances in Spatial Databases, Lecture Notes in Computer Science*, 525, Springer-Verlag, 1991.

[142] Hoffmann, C.M., The problems of accuracy and robustness in computational geometry, *IEEE Computer*, 1989, 22, 3, pp.31-41.

[143] Hoffmann, C., *Geometric and solid modeling: an introduction*, Morgan Kaufmann, San Mateo, California, 1989, The Morgan Kaufman Series in Computer Graphics and Geometric Modeling.

[144] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., Surface reconstruction from unorganized points, *Proceedings SIGGRAPH '92 Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH*, 1992, pp.71-78.

[145] Hoppe, H., Progressive meshes, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '96)*, 1996, pp.99-108.

[146] Hoppe, H., View-dependent refinement of progressive meshes, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '97)*, 1997.

[147] Imai, H., Iri, M., Computational-geometric methods for polygonal approximations of a curve, *Computer Vision, Graphics and Image Processing*, 36, 1986, pp.31-41.

[148] Imai, H., Iri, M., An optimal algorithm for approximating a piecewise linear function, *Journal of Information Processing*, 9, 3, 1986, pp.159-162.

[149] Imai, H., Iri, M., Polygonal approximation of a curve - formulations and algorithms, *Computational Morphology*, G.T. Toussaint (Ed.), *Machine Intelligence and Pattern Recognition*, 6, North-Holland, 1988, pp.71-86.

[150] Jenson, S.K., Dominique, J.O., Extracting topographic structure from digital elevation data for geographic information system analysis, *Photogrammetric Engineering and Remote Sensing*, 54, 1988, pp.1593-1600.

[151] Joe, B., Construction of three-dimensional Delaunay triangulations using local transformations, *Computer Aided Geom. Design*, 8, 2, 1991, pp.123-142.

[152] Jones, C., Cartographic name placement with Prolog, *IEEE Computer Graphics and Applications*, 9, 5, 1989, pp.36-47.

[153] Jones, C.B., Data structures for three-dimensional spatial information systems in geology, *International Journal of Geographic Information Systems*, 3, 1, 1989, pp.15-32.

[154] Jones C.B., Bundy, G.L., Ware, J.M., Map generalization with a triangulated data structure, *Carthography and Geographic Information Systems*, 22, 4, 1995, pp.317-331.

[155] Jorge, J.A., Vaida, D., A fuzzy relational path algebra for distances and directions, *Proceedings ECAI-96 Workshop on Representation and Processing of Spatial Expressions*, Budapest, Hungary, 1996.

[156] Kainz, W., Spatial relationships - Topology versus order, *Proceedings 4th International Symposium on Spatial Data Handling*, pp.814-819, Zurich, Switzerland, July 1990.

[157] Kainz, W., Egenhofer, M.J., Greasley, J., Modeling spatial relations and operations with partially ordered sets, *International Journal of Geographic Information Systems*, 7, 3, 1993, pp.215-225.

[158] Katz, M.J., Overmars, M.H., and Sharir, M., Efficient hidden surface removal for objects with small union size, *Proceedings 7th ACM Symposium on Computational Geometry*, ACM Press, New York, 1991, pp.31-40.

[159] Keppel, E., Approximating complex surfaces by triangulation of contour lines, *IBM Journal of Research and Development*, 19, 1, 1975, pp.2-11.

[160] Kirkpatrick, D.G., Efficient computation of continuous skeletons, *Proceedings 20th IEEE Symposium on Foundations of Computer Science*, 1979, pp.18-27.

[161] Kirkpatrick, D.G., Optimal search in planar subdivisions, *SIAM Journal of Computing*, 12, 1, 1983, pp.28-35.

[162] Klein, R., Straßer, W., Generation of multiresolution models from CAD data for real time rendering, *Theory and Practice of Geometric Modeling*, W. Straßer, R. Klein, R. Rau (Eds.), Springer-Verlag, 1996.

[163] Koenderik, J.J., van Doorn, A.J., Local features of smooth shapes: ridges and courses, in *Geometric Methods in Computer Vision II*, vol. 2031, SPIE 1993, pp.2-13.

[164] Koenderik, J.J., van Doorn, A.J., Two-plus-one-dimensional differential geometry, *Pattern Recognition Letters*, 15, 1994, pp.439-443.

[165] Kong, T.Y., Rosenfeld, A., Digital topology: introduction and survey, *Computer Vision, Graphics and Image Processing*, 48, 1989, pp.357-393.

[166] Kriegel, H.P., Brinkhoff, T., Schneider, R., The combination of spatial access methods and Computational Geometry in Geographic Database Systems, *Proceedings 2nd Symposium on Spatial Databases*, 1991, pp.5-21.

[167] Kriegel, H.P., Brinkhoff, T., Schneider, R., An efficient map overlay algorithm based on Spatial Access Methods and Computational Geometry, *Proceedings International Workshop on DBMSs for Geographical Applications*, Capri, Italy, May 1991, pp.16-17.

[168] Kučera, L., Melhorn, M., Preis, B., Schwarzenecker, E., Exact algorithms for a geometric packing problem, *Proceedings 10th Symposium on Theoret. Aspects of Computer Science*, *Lecture Notes in Computer Science*, 665, Springer-Verlag, 1993, pp.317-322.

[169] Kweon, I.S., Kanade, T., Extracting topological terrain features from elevation maps, *Computer Vision, Graphics and Image Processing*, 59, 2, 1994, pp.171-182.

[170] Lanthier, M., Maheshwari, A., Sack, J.R., Approximating weighted shortest paths on polyhedral surfaces, *Proceedings ACM Symposium on Computational Geometry*, Nice, France, 1997, pp.274-283.

[171] Laurini, R., Thompson, D., *Fundamentals of Spatial Information Systems*, Academic Press, London, 1992.

[172] Lawson, C.L., Software for C^1 surface interpolation, *Mathematical Software III*, J.R. Rice (Ed.), Academic Press Inc., 1977, pp.161-164.

[173] Lee, D.T., Lin, A.K., Generalized Delaunay triangulation for planar graphs, *Discrete and Computational Geometry*, 1, 1986, pp.201-217.

[174] Lee, J., A drop heuristic conversion method for extracting irregular networks for digital elevation models, *Proceedings GIS/LIS '89*, Orlando, FL, USA, 1989, pp.30-39.

[175] Lee, J., Analyses of visibility sites on topographic surfaces, *International Journal of Geographical Information Systems*, 5, London, Taylor & Francis, 1991, pp.413-429.

[176] Li, Z., Openshaw, S., Algorithms for automated line generalization based on a natural principle of objective generalization, *International Journal of Geographical Information Systems*, 6, 5, 1992, pp.373-389.

[177] Lienhardt, P., Topological Models for Boundary Representations: a Comparison with n-dimensional Generalized Maps, *Computer Aided Design*, 23, 1, 1991, pp.59-82.

[178] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N., Turner, G.A., Real-Time, Continuous Level of Detail Rendering of Height Fields, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '96)*, ACM Press, New Orleans, LA, USA, 1996, pp.109-118.

[179] Liotta, G., Preparata, F.P., Tamassia, R., Robust proximity queries: an illustration of degree-driven algorithm design, *Proceedings 13th ACM Symposium on Computational Geometry*, 1997, pp.156-165

[180] Livnat, Y., Shen, H.W., Johnson, C.R., A near optimal isosurface extraction algorithm using the span space, *IEEE Transactions on Visualization and Computer Graphics*, 2, 1996, pp.73-84.

[181] Lounsbery, M., DeRose, T.D., Warren, J., Multiresolution analysis of surfaces of arbitrary topological type, *ACM Transactions on Graphics*, 16(1), 1997, pp.34-73.

[182] Lukatela, H., Hipparchus data structures: points, lines and regions in spherical Voronoi grid, *Proceedings Auto-Carto 9*, 1989, pp.164-170.

[183] Lundell, A.T., Weingram, S., *The Topology of CW Complexes*, Van Nostrand Reinhold Comp., 1969.

[184] Magillo, P., De Floriani, L., Maintaining multiple levels of detail in the overlay of hierarchical subdivisions, *Proceedings 8th Canadian Conference on Computational Geometry*, 1996, pp.190-195.

[185] Magillo, P., Puppo, E., Algorithms for Parallel Terrain Modelling and Visualisation, Chapter 16 in *Parallel Processing Algorithms for GIS*, R.G. Healey, S. Dowers B.M. Gittings, M.J. Mineter (Eds.), Taylor & Francis, 1998.

[186] Maheshwari, A., Morin, P., Sack, J.-R., Progressive TINs: algorithms and applications, *Proceedings 5th ACM Workshop on Advances in Geographic Information Systems*, November 15-16, 1997, Las Vegas (Nevada).

[187] Mäntylä, M., *An Introduction to Solid Modeling*, Computer Press, 1988.

[188] Mark, D.M., Topological properties of geographic surfaces: applications in computer carthography, *Proceedings 1st International Advanced Study Symposium on Topological Data Structures for GIS*, 5, G. Dutton (Ed.), Harvard, 1994.

[189] Mata, C.S., Mitchell, J.S.B., A new algorithm for computing shortest paths in a weighted planar subsivision, *Proceedings ACM Symposium on Computational Geometry*, Nice, France, June 1997, pp.264-273.

[190] Mc Cormack, J.E., Gahegan, M.N., Roberts, S.A., Hogg, J., Hoyle, B.S., Feature-based derivation of drainage networks, *International Journal of GIS*, 7(3), 1993, pp.263-279.

[191] McKenna, M., Worst case optimal hidden surface removal, *In ACM Transactions on Graphics*, 6, ACM Press, New York, 1987, 19-28.

[192] McMaster, R., Automated line generalization, *Cartographica*, 24, 1984, pp.74-111.

[193] Miller, C., LaFlamme, R.A., 1958, The digital terrain model - theory and applications, *Photogrammetric Engineering*, 24(3), pp.433-442.

[194] Misund, G., Varioscale TIN based surfaces, *Proceedings 7th Int. Symp. on Spatial Data Handling*, August 12-16, 1996, Delft (NL), pp.6.35-6.46.

[195] Mitchell, J.S.B., Mount, D.M., Papamitridiou, C.H., The discrete geodesic problem, *SIAM Journal of Computing*, 16, 1987, pp.647-668.

[196] Mitchell, J.S.B., Papamitridiou, C.H., The weighted region problem: finding shortest paths through a weighted planar subsivision, *Journal of the ACM*, 38, 1, January 1991, pp.18-23.

[197] Mitchell, J.S.B., Suri, S., A survey of Computational Geometry, M.O. Ball et al. (Eds.), *Handbooks in Operations Research and Management Science*, Vol. 7, Chapter 7, Elsevier Sciance B.V., 1995, pp.439-458.

[198] Molenaar, M., Single valued vector maps - a concept in GIS, *Geo-Informationssysteme*, 2, 1, 1989.

[199] Mortenson, M.E., *Geometric Modeling*, John Wiley, New York 1985.

[200] Morris, D.G., Heerdegen, R.G., Automatically derived catchment boundaries and channel networks and their hydrological applications, *Geomorphology*, 1, 1988, pp.131-141.

[201] Mount, D.M., On finding shortest paths on convex polyhedra, *Technical Report* 1495, Department of Computer Science, University of Maryland, Baltimore, MD, 1985.

[202] Mulmuley, K., *Computational Geometry: an Introduction through Randomized Algorithms*, Prentice Hall, 1994.

[203] Nagy, G., Terrain Visibility, *Computers and Graphics*, 18, 6, 1994.

[204] Näher, S., *The LEDA user manual*, Version 3.1 (Jan. 16, 1995). Available by anonymous ftp from ftp.mpi-sb.mpg.de in directory /pub/LEDA.

[205] Naylor, B., Amanatides, J., Thibault, W., Merging BSP-trees yelds polyhedral set operations, *Computer Graphics (Proceedings SIGGRAPH 1990)*, 24, 1990, F. Baskett (editor), pp.115-124.

[206] Nielson, G., Scattered data modeling, *IEEE Computer Graphics and Applications*, 13, 1, 1993, pp.60-70.

[207] Nievergelt, J., Preparata, F.P., Plane sweep algorithms for intersecting geometric figures, *Communications of the ACM*, 25, 10, 1982, pp.739-747.

[208] Nurmi, O., A fast line-sweep algorithm for hidden line elimination, *BIT*, 25, 1985, pp.466-472.

[209] OpenInventor Architecture Group, *Inventor Mentor: OpenInventor Reference Manual*, Addison Wesley, 1994.

[210] Orenstein, J., An algorithm for computing the overlay of k-dimensional spaces, *Proceedings 2nd Symposium on Advanced in Spatial Databases*, *Lecture Notes in Computer Science*, 525, O.Günter, H.J.Schek (Eds.), 1991, pp.381-400.

[211] Overmars, M., and Sharir, M., A simple output-sensitive algorithm for hidden surface removal, *ACM Transactions on Graphics*, 11, ACM Press, New York, 1992, pp.1-11.

[212] Overmars, M.H., Designing the computational geometry algorithms library CGAL, in *Applied Computational Geometry* (Proc. WACG'96), *Lecture Notes in Computer Science*, 1997, Springer-Verlag.

[213] Palazzi, L., Snoeyink, J., Counting and reporting red/blue segment intersections, *Lecture Notes in Computer Science*, 709, Springer Verlag, New York, F.Dehne, J.R.Sack, N.Santoro, S.Whitesides (Eds.), 1993, pp.530-540.

[214] Persson, J., Jungert, E., Generation of multi-resolution maps from run length-encoded data, *International Journal of Geographical Information Systems*, 6, 6, 1992, pp.497-510.

[215] Petrie, G., Kennie, T.J.M., (Editors), *Terrain Modelling in Survey and Civil Engineering*, Whittles Publishing - Thomas Telford, London, 1990, pp.112-127.

[216] Petrie, G., Modelling, interpolation and contouring procedures, in *Terrain Modelling in Survey and Civil Engineering*, G. Petrie and T.J.M. Petrie (Eds.), Whittles Publishing - Thomas Telford, London, 1990, pp.112-127.

[217] Peucker, T.K., Douglas, D.H., Detection of surface-specific points by local parallel processing of discrete terrain elevation data, *Computer Graphics and Image Processing*, 4, 1975, pp.375-387.

[218] Pigot, S., Generalized singular 3-cell complexes, *Proceedings 6th International Symposium on Spatial Data Handling*, Edinburgh, Scotland, 1994, pp.89-111

[219] Pilouk, M., Tempfli, K., Molenaar, M., A tetrahedron-based 3D vector data model for geoinformation, *Advanced Geographic Data Modelling - Spatial Data Modelling and Query Languages for 2D and 3D Applications*, M. Molenaar, S. De Hoop (Eds.), Publication on Geodesy - New Series, 40, 1994, pp.129-140.

[220] Pratt, W.K., *Digital Image Processing*, John Wiley & Son, 1978.

[221] Preparata, F.P., and Shamos, M.I., *Computational Geometry: an Introduction*, Springer Verlag, Berlin, 1985.

[222] Preparata, F.P., Vitter, J.S., A simplified technique for hidden-line elimination in terrains, *Proceedings STACS'92*, Paris, A.Finkel, M.Jantzen (Eds.), *Lecture Notes in Computer Science*, 577, Springer Verlag, Berlin, 1992, pp.135-144.

[223] Puppo, E., Dettori, G., Towards a Formal Model for Multiresolution Spatial Maps, in M.J. Egenhofer, J.R. Herring (Eds.), in *Advances in Spatial Databases*, *Lecture Notes in Computer Science*, 951, Springer-Verlag, 1995, pp.152-169.

[224] Puppo, E., Variable resolution terrain surfaces, *Proceedings Canadian Conference on Computational Geometry*, Ottawa (Canada) 12-15 August, 1996; appeared in longer version as Variable resolution triangulations, *Computational Geometry Theory and Applications*, 11 (3–4), 1998, pp. 219–238.

[225] Puppo, E., Marzano, P., Discrete visibility problems and graph algorithms, *International Journal of Geographic Information Science*, 11, 2, 1997, pp.139-162.

[226] Puppo, E., Scopigno, R., Simplification, LOD, and multiresolution - Principles and applications, *Eurographics'97 Tutorial*, 1997.

[227] Rajan, V.T., Optimality of the Delaunay triangulation in R^d, *Discrete and Computational Geometry*, 12, 1994, pp.189.

[228] Raper, J., (Ed.), *Three Dimensional Applications in Geographic Information Systems*, Taylor & Francis, 1989.

[229] Reif, J.H., and Sen, S., An efficient output-sensitive hidden-surface removal algorithm and its parallelization, *Proceedings 4th ACM Symposium on Computational Geometry*, Urbana, ACM Press, New York,1988, pp.193-200.

[230] Rippa, S., Minimal roughness property of Delaunay triangulation, *Computer Aided Geometric Design*, 7, 1990, pp.489-497.

[231] Rippa, S., Adaptive approximations by piecewise linear polynomials on triangulations of subsets of scattered data, *SIAM Journal on Scientific and Statistic Computing*, 13, 1, 1992, pp.1123-1141.

[232] Robinson, A.H., Sale, R.D., Morrison, J.L., Muehrcke, P.C., *Elements of Cartography*, John Wiley, New York, 5th edition, 1984.

[233] Rokne, J.G., Wyvill, B., Wu, X., Fast line scan-conversion, *ACm Transactions on Graphics*, 9, 1990, pp.376-388.

[234] Rosenfeld, A., Kak, A., Digital Picture Processing, *Computer Science and Mathematics*, Vol. 1 and 2, 2nd edition, Academic Press, 1982.

[235] Roussopoulos, N., Leifker, D., Direct spatial search on pictorial databases using packed R-trees, *Proceedings SIGMOD Conference*, 1985, pp.17-31.

[236] Ruppert, J,, Seidel, R., On the difficulty of tetrahedralizing 3-dimensional non-convex polyhedra, *Proceedings 5th ACM Symposium on Computational Geometry*, 1989, pp.380-393.

[237] Saalfeld, A., Joint triangulations and triangulation maps, *Proceedings Third Symposium on Computational Geometry*, Waterloo, Canada, June 1987, pp.195-204.

[238] Saalfeld, A., Conflation: automatic map compilation, *International Journal of Geographic Information Systems*, 2,3, 1988, pp.217-228

[239] Saalfeld, A., Delaunay edge refinements, *Proceedings 3rd Canadian Conference on Computational Geometry*, 1991, pp.33-36.

[240] Saalfeld, A., Comflation: automated map compilation, *Technical Report* CAR-TR-670, Center for Automation Research, May 1993.

[241] Saalfeld, A., Map generalization as a graph drawing problem, in R. Tamassia and I. G. Tollis (Eds.), *Graph Drawing (Proceedings GD '94)*, *Lecture Notes in Computer Science*, 894, Springer-Verlag, 1995, pp.444-451.

[242] Samet, H., *The design and analysis of spatial data structures*, Addison-Wesley, Reading, MA, 1990.

[243] Samet, H., *Applications of spatial data structures*, Addison-Wesley, Reading, MA, 1990.

[244] Samet, H., Object-Based and Image-Based Representations of Objects by their Interiors, in Advances in Image Understanding - A Festschrift for Azriel Rosenfeld, (K. Bowyer and N. Ahuja, Eds.), IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 316-332.

[245] Scarlatos, L.L., An automatic critical line detector for digital elevation matrices, *Proceedings 1990 ACSM-ASPRS Annual Convention*, 2, Denver, CO, 1990, pp.43-52.

[246] Scarlatos, L.L., Pavlidis, T., Hierarchical triangulation using cartographic coherence. *CVGIP: Graphical Models and Image Processing*, 1992, 54, 2, pp.147-161.

[247] Schmitt, A., Time and space bounds for hidden line and hidden surface algorithms, *Proceedings Eurographics'81*, North Holland, Amsterdam, 1981, pp.43-56.

[248] Schubert, L., Wang, C.A., An optimal algorithm for constructing the Delaunay triangulation of a set of line segments, *Proceedings 3rd ACM Symposium on Computational Geometry*, Waterloo, June 1987, pp.223-232.

[249] Schroeder, W.J., Zarge, J.A., Lorensen, W., Decimation of triangle mesh, *ACM Computer Graphics*, 26, 2, July 1992, pp.65–70.

[250] Sellis, T., Roussopoulos, N., Faloutsos, C., The R^+-tree: a dynamic index for multidimensional objects, *Proceedings 13th International Conference on Very Large Data Bases* (VLDB), 1987, pp.507-518.

[251] Sharir, M., Shorr, A., On shortest paths on polyhedral spaces, *SIAM Journal of Computing*, 15, 1986, pp.193-215.

[252] Sharir, M., The shortest watchtower and related problems for polyhedral terrains, *Information Processing Letters*, 29, 1988, pp.265-270.

[253] Silva, C.T., Mitchell, J.S.B., Kaufman, A.E., Automatic generation of triangular irregular networks using greedy cuts, *Proceedings IEEE Visualization'95*, 1995, pp.201-208.

[254] Skidmore, A.K., Terrain position as mapped from gridded digital elevation model, *International Journal of Geographical Information Systems*, Taylor & Francis, 4, 1, 1990, pp.33-49.

[255] Sloan, K.R., Hrechanyk, L.D., Surface reconstruction from sparse data, *Proceedings Conference on Pattern Recognition and Image Processing*, 1981, pp.45-48.

[256] Snoeyink, J., van Kreveld, M., Linear-time reconstruction of Delaunay triangulations with applications, *Proceedings 5th European Symposium on Algorithms*, 1997.

[257] Southard, D.A., Piecewise linear surface models from sampled data, *Proceedings Computer Graphics International 91*, Boston, MA, 22-28 June, 1991.

[258] Suri, S., A linear time algorithm for minimum link paths inside a simple polygon, *Computer Vision, Graphics and Image Processing*, 35, 1986, pp.99-110.

[259] Stollnitz, E.J., DeRose, T.D., Salesin, D.H., Wavelets for computer graphics: a primer, Part I, *IEEE Computer Graphics and Applications*, May 1995, pp.76-84.

[260] Stollnitz, E.J., DeRose, T.D., Salesin, D.H., Wavelets for computer graphics: a primer, Part II, *IEEE Computer Graphics and Applications*, July 1995, pp.75-85.

[261] Tamassia, R., (Editor), Strategic Directions in Computational Geometry Working Group Report, *ACM Computing Surveys*, 28, 4, 1996.

[262] Taubin, G., Rossignac, J., Geometric compression through topological surgery, *IBM Research Report*, RC-20340, January 1996.

[263] Tobler, W.R., A continuous transformation useful for districting, *Ann. New York Acad. of Science*, 219, 1973, pp.215-220.

[264] Tsai, V.J.D., Delaunay triangulations in TIN creation: an overview and a linear-time algorithm, *International Journal of GIS*, 7(6), 1993, pp.501-524.

[265] Turk, G., Re-Tiling Polygonal Surfaces, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '92)*, 26, 2, 1992, pp.55-64.

[266] van Bemmelen, J., Quak, W., van Hekken, M., van Oosterom, P., Vector vs. raster-based algorithms for cross country movement planning, *Proceedings AUTO-CARTO 11*, Minneapolis, 1993, pp.309-317.

[267] Veltkamp, R.C., 3D Computational Morphology, *EUROGRAPHICS '93*, pp. 115-127, 1993.

[268] van Kreveld, M., On quality paths on polyhedral terrains, *Proceedings IGIS'94: Geographic Information Systems*, Nievergelt, J., Roos, T., Schack, H.J., Widmayer, P. (Eds.), *Lecture Notes in Computer Science*, 884, Springer-Verlag, 1994, pp.113-122.

[269] van Kreveld, M., Efficient methods for isoline extraction from a TIN, *International Journal of Geographic Information Science*, 10, 5, 1996, pp.523-540.

[270] van Kreveld, M., Digital elevation models and TIN algorithms, *Notes for CISM Advanced School on Algorithmic Foundations of GIS*, 1996.

[271] van Kreveld, M., Variations on sweep algorithms: efficient computation of extended viewsheds and class intervals, *Proceedings Symposium on Spatial Data Handling*, 1996, pp.13A.15-13A.27.

[272] van Kreveld, M., van Oostrum, R., Bajaj, C., Contour trees and small seed sets for isosurface traversal, *Proceedings 13th Annual ACM Symposium on Computational Geometry*, 1997, pp.212-219.

[273] van Oosterom, P., A modified binary space partition for geographic information systems, International Journal of Geographical Information Systems, 4, 2, 1990, pp.133-146.

[274] van Oosterom, P., *Reactive Data Structures for Geographic Information Systems*, Oxford University Press, 1993.

[275] van Oosterom, P., Schaukelaars, V., The development of an interactive multi-scale GIS, *International Journal of GIS*, 9, 5, 1995, pp.489-507.

[276] Vincent, I., Soille, P., 1991, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13,pp. 583-598

[277] *The Virtual Reality Modeling Language Specification - Version 2.0*, August 1996, http://vag.vrml.org/.

[278] Von Herzen, B., Barr, A.H., Accurate triangulations of deformed, intersecting surfaces, *ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '87)*, 21, 4, July 1987, pp.103-110.

[279] Wagner, F., Wolff, A., An efficient and effective approximation algorithm for the map labeling problem, *Proceedings 3rd European Symposium on Algorithms*, *Lecture Notes in Computer Science*, 979, 1995, pp.420-433.

[280] Wagner, F., Wolff, A., Map labeling heuristics: provably good and practically useful, *Proceedings 11th Annual ACM Symposium on Computational Geometry*, 1995, pp.109-118.

[281] Ware, J.M., Jones, C.B., Bundy, G.L., A triangulated spatial model for cartographic generalization of areal objects, in A. Frank and W. Kuhn (Eds.), *Spatial Information Theory*, *Lecture Notes in Computer Science*, 988, Springer-Verlag, 1995, pp.173-191.

[282] Watson, D.F., Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, *The Computer Journal*, 24, 1981, pp. 728-746.

[283] Watson, L.T., Laffey, T.J., Haralick, R.M., Topographic classification of digital image intensity surfaces using generalized splines and the discrete cosine transformation, *Computer Vision, Graphics and Image Processing*, 29, 1985, pp.143-167.

[284] Webb, H., Creation of digital terrain models using analytical photogrammetry and their use in civil engineering, *Terrain Modelling in Survey and Civil Engineering*, G. Petrie and T.J.M. Petrie (Editors), Whittles Publishing - Thomas Telford, London, 1990, pp.73-84.

[285] Weiler, K., Edge-based Data Structures for Solid Modeling in a Curved-surface Environment, *IEEE Computer Graphics and Applications*, 5, 1, January 1985, pp.21-40.

[286] Weibel, R., Hellen, M., A framework for digital terrain modeling, *Proceedings 4th International Conference on Spatial Data Handling*, Zurich, 1990, pp.219-229.

[287] Weibel, R., A typology of constraints for line simplification *Proceedings 7th International Symposium on Spatial Data Handling* (SDH'96), 1996, pp.9A.1-9A.14

[288] Wernecke, J., *The Inventor mentor: programming Object-oriented 3D graphics with Open Inventor*, Addison Wesley, 1994.

[289] Wolf, G.W., Metric surface networks, *Proceedings 4th International Symposium on Spatial Data Handling*, Zuerich, 1990, pp.844-846.

[290] Woo, T.C., A Combinatorial Analysis of Boundary Data Structure Schemata, *IEEE Computer Graphics and Applications*, 5, 3, 1985, pp.19-27.

[291] Worboys, M.F., A generic model for planar geographic objects, *International Journal of Geographical Information Systems*, 6, 5, 1992, pp.353-372.

[292] Worboys M.F., Bokafos, P, A canonical model for a class of areal spatial objects. Advances in Spatial Database (SSD93), D. Abel, B.C. Ooi (Eds.), Lecture Notes in Computer Science, Springer-Verlag, 1993, pp.36-52

[293] Xia, J.C., Varshney, A., Dynamic View-dependent simplification for polygonal models, *Proceedings IEEE Visualization '96*, S. Francisco, CA, R. Yagel and G. Nielson (Eds.), 1996, pp.327-334.

[294] Yu, S., Van, M., Snoeyink, J., Drainage queries in TINs: from local to global and back again, *Proceedings Symposium on Spatial Data Handling (SDH'96)*, 1996, pp.13A.1-13A.14.

[295] Zhu, B., Computing the shortest watchtower of a polyhedral terrain in $O(n \log n)$ time, Computational Geometry - Theory and Applications, 8, 1997, pp.181-193.

[296] Zoraster, S., Integer programming applied to the map label placement problem, *Cartographica*, 22, 3, 1986, pp.16-27.