Spatial Data Structures in Geographic Information Systems

P.J.M. van Oosterom
Department of Computer Science, University of Leiden,
TNO Physics and Electronics Laboratory, The Hague,
Email: oosterom@hlerul5.bitnet.

August 31, 1988

Abstract

This document gives an overview of Geographic Information Systems (GISs) with respect to the Data Model. We introduce a classification of the types of data that one encounters in GISs. The requirements for the GIS Data Model, that we are developing, are summarized in a short list. The known solutions for storing the spatial data are described together with a new method. We classify and discuss the data structures for storing topological data. Finally, some additional topics and open problems are indicated.

1 Introduction

In the past few years there has been a growing interest in the application of Geographic Information Systems. There are many applications that use the GIS technology. Some of them are: Hydrographic, Topographic and Cadastral Mapping; Demography and Environmental Planning; Automated Mapping / Facility Management (AM/FM); Command, Control and Communication Systems; War gaming and other Simulations based on Terrain Data; Car or Ship Navigation Systems.

One of the main reasons behind this development is the availability of relatively inexpensive hardware like powerful minicomputers and workstations, graphics displays, scanners, digitizers, plotters and printers. The advent of cheap hardware encouraged the latent demand for GISs with new capabilities. The geographic part of a system might be coupled with other parts of the information system. For example, in the case of a navigation or a simulation system. Another major advantage of a GIS is that, although a map is presented on a display in the same way as the traditional paper map, the operator (end-user) can *interact* with the system. To make it both possible and efficient to answer queries from the operator, the GIS has to be based on an appropriate data structure. However, most existing systems lack these data structures.

The purpose of this research is to create a model, which an application programmer can use to develop a specific Geographic Information System. The model must cover aspects such as the integrated storage and presentation of graphical and non-graphical data and the interaction with the operator. The backbone of this model should be a spatial data structure which must allow the system to give fast respond to queries.

In a Geographic Information System the integrated storage of graphical and non-graphical data plays an important role. To present this data, graphical packages like GKS [4] or PHIGS [28] can be used. However, their use for the storage of data is limited: only graphical data can be stored; only storage at execution time (or storage in a metafile) is possible; very limited capabilities are available for structuring the data.

Database packages (Network, Hierarchical or Relational Data Base Management Systems; DBMSs [11,51]) exist for long-term and structured storage. The drawback of these packages is that they do not handle geographical data very well. Queries like: 'which cities with a population greater than 100.000 lie within 10 kilometers of the river Rhine', are in general not easily answered by database packages. Therefore,

research is necessary to develop a data model that does not have the restrictions of the graphical and database packages mentioned above.

2 Data in a GIS

The entities in a GIS are based on three different types of data: *geometrical, topological* and *application* data. Geometrical data have a quantitative nature and are used to represent coordinates, line equations, etc. Basically, there are two formats: raster and vector format. The vector format has three subtypes: point, polyline and polygon. Henceforth these subtypes are called geometric primitives.

Topological data describe the relationships between the geometrical data. Examples of topological data are: relationships that describe which points define a line, which areas are neighbours of each other, which polylines form a network (for representing a network of roads). Topological data is not always stored explicitly.

Application data are non-graphical data which are somehow related to the data described above; e.g. the name and capacity of a road (represented by a polyline). Application data may be any kind of data that can be found in the traditional databases.

The remainder of this section explains why this document concentrates on the vector format instead of the raster format for the geometrical data. Note that this discussion is about the internal representation and not about the display system. An advantage of the raster format is that it is appropriate for data capture techniques like remote sensing (satellites), scanning of existing documents (maps or aerial photographs) and image processing in general, which all deal with raster data. The raster format has some good spatial properties, inherent to the direct addressing of the pixels. Some applications of these properties are:

- *select* a certain (rectangular) region by using all the pixels with the x and y indices in proper range;
- find the *neighbours* of a pixel by using all the pixels which have indices one lower or higher in comparison with the original pixel;
- if each polygon is represented by a set of pixels with the same colour and every polygon has a different colour, then the *point-in-polygon* test is established by determining the colour of the pixel (point);
- if two maps are based on the same raster then *map overlay* is performed by a simple pixel by pixel algorithm (this algorithm is also well suited for hardware implementation).

The fact that a raster algorithm is simple does not mean that it is faster than a more complex vector algorithm, because raster data have the tendency to be very voluminous. One reason for not conducting research on raster data structures in GIS is that the raster format already possesses good spatial properties. More important reasons for concentrating the research on vector data structures are the serious disadvantages of the raster format. Different sources of raster data use different rasters (sometimes a pixel does not represent a square but a rectangle). What is worse, a rectangular raster may not even be rectangular when using a different projection. The geometric data should not depend on a specific projection. That is, the *geographic coordinate system* must be used. The main drawback of the raster format is that it deals with digital images and not with parts of geographic entities that can be displayed and manipulated individually. Therefore, it is not possible to link the geometric data, in a natural way, with the other data types of the geographical entities.

In spite of these arguments a GIS should possess the capability to use raster data as well as vector data, because a user often has no control over the format in which his data is delivered. Raster data can be incorporated in a vector based GIS by treating these data as colour or texture function on polygons. For example a satellite image may be projected on a Digital Elevation Model (DEM, see subsection 3.7).

3 Operations in a GIS

This section describes operations that are used by an operator of a GIS, possibly by way of an application program. We identify two groups of operations. The first group consists of the operations that must be present in every GIS, these are called fundamental operations. The second group consists of the operations specific to a certain application, the additional operations. All operations are meant to be used in an interactive mode by the operator, possibly through an application program. This section discusses all the

fundamental operations and a few examples of operations from the second group. The spatial data structure (section 5) should allow efficient implementation of the operations.

There are three types of fundamental operations: *display* the map, *select* entities from the map (pick) and *spatial calculations* with 1 or 2 geographic entities as operands. These operations will be discussed in subsections 3.1 through 3.3. Some additional operations will be treated in subsections 3.4 through 3.6. The operations on Digital Elevation Models form a separate group and will be discussed in subsection 3.7.

3.1 Display Map

The display map operation is the most essential GIS operation and seems easy; just draw what is stored in the database. However, this is not true, first a number of more complicated steps are made. The *projection* type is chosen and there are other transformations that might also be of interest. In a navigation system for example, the display can be *north-up* or *head-up*.

A GIS contains several layers of information, so it has to be decided which layers will be used. Furthermore, the region which has to be displayed and the scale to be used must be selected.

After the map is drawn the operator may want to look at an adjacent part of the map, this is called *panning*. It is also possible that the operator wants to take a closer look at a part of the map, the *zoom-in* operation. In this case not only the objects are enlarged on the display, but also more details are drawn. The reverse operation of zoom-in is of course *zoom-out*, where details are removed.

In many situations it is possible to draw different kinds of maps using the same data or derivations thereof. To produce a thematic map one could use [39]: a choropleth, a dot-chart, an iso-line map, a trend surface, a PRISM chart [20], etc., to visualize the theme. Some GISs consist of a powerful display operation only; for example IDECAP [53].

3.2 Select Entities

There is a strong relationship between the display map and the select operation. On the one hand, the selected entities are be displayed on the screen. On the other hand, entities may be selected from those displayed on the screen.

The selection queries in a GIS are the same as in a traditional DBMS, except that the user may also want to pose *spatial* queries and *hybrid* queries That is, a combination of the traditional and the spatial queries. Traditional queries are stated in an alpha-numerical way using a query language like Structured Query Language (SQL [11]). Selection criteria may be based on spatial calculations, see subsection 3.3. For instance: 'select all polygons with area greater than 2 square kilometers'.

Spatial queries can be made more easily by using a *pick* input device. There are three important issues related to a pick device:

- Which types of geometric primitives are to be selected: points, polylines or polygons?
- What kind of geometric primitive is used to perform the select? This object is called the *pick primitive*. The two most frequently used are point and rectangle.
- How large should the pick *aperture* be? That is, how large is the maximum distance between the pick primitive and the primitives which are to be selected.

Note that other pick primitives might be useful. For example a GIS, for planning a route of a new road, the pick primitive is polyline. This polyline represents the new road and the primitives that are to be selected by the pick are the polygons representing plots of ground with their owners.

3.3 Spatial Calculations

This subsection describes spatial calculations of general interest: distance, circumference, area and center of gravity. These calculations are used in the select or display operations. Some examples: the area of a polygon is used to convert absolute data (e.g. number of inhabitants of a region) into relative data (population density); the center of gravity is used for the position of a label 'in' a polygon. The *Euclidean* distance between point $p_1 = (x_1, y_1)$ and point $p_2 = (x_2, y_2)$ is:

$$D_E(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Another useful distance function is the *City block* distance:

$$D_C(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$$

Formulas for calculating the shortest distance between a point and a polyline or a polygon are a little more complicated. The same applies to the distance between two polylines, two polygons and also between a polyline and a polygon. The circumference of a polygon defined by points p_1 through p_n (with coordinates (x_i, y_i) for i from 1 to n and $p_{n+1} = p_1$) is:

$$C = \sum_{i=1}^{n} D_E(p_i, p_{i+1})$$

The area can be computed by accumulating a running balance of triangular areas, resulting in the formula (when the nodes are numbered in counter clockwise order):

$$A = \frac{1}{2} \sum_{i=1}^{n} (x_i y_{i+1} - y_i x_{i+1})$$

If the order in which the nodes are numbered is unknown, the true area is the absolute value of A. The center of gravity (x_c, y_c) is computed similarly:

$$x_c = \frac{1}{6A} \sum_{i=1}^{n} (x_i + x_{i+1})(x_i y_{i+1} - y_i x_{i+1})$$

$$y_c = \frac{1}{6A} \sum_{i=1}^{n} (y_i + y_{i+1})(x_i y_{i+1} - y_i x_{i+1})$$

3.4 Combining Maps

The ability to combine different maps is one of the main strengths of a GIS. The first variant of this operation is the *visual* combination of different maps using the display map operation with the right projection and scale. An example of this is: first draw a polygonal map with soil types, then draw the network map of roads and finally draw the point map which represents the cities. A special type of the visual combine-map operation is the display of a map projected on a DEM.

A problem arises when two polygonal maps, covering the same region, have to be combined. The second map could be drawn in a semi-transparent mode, but this is not always a satisfying solution.

The second variant of map combination operates on two polygonal maps. It first computes the resulting map and then displays it. This is the *polygon-overlay* problem and several algorithms are described in the literature [21,16]. Because polygon-overlay requires time consuming computations, it might be useful to store frequently used map overlays [19]. Note that to solve the query 'compute the total area of sand grounds that lie within 2 kilometers of a road', the polylines representing roads have to be converted to a polygon map before an overlay with the soil type map can be produced.

3.5 Network Analysis

For map layers with a network topology, see section 6, there are several analysis operations. Some examples are [32,34,31]:

- *shortest path:* Calculate the shortest path between an origin and destination. The best path problem is a modification hereof which tries to minimize the sum of the weight values, rather than the geometric length, of the edges.
- *location of service center:* Determine the best location of a service center, e.g. a post office or a bus stop. Try to minimize the total distance between the service center and the homes of the inhabitants in that region.
- *travelling salesman:* Determine the best or shortest route for a salesman who has to visit a number of cities.

The problems described above and other Operation Research problems may require a lot of computation, especially as the size of the problem increases, e.g. the number of cities a salesman has to visit. Therefore, sometimes an almost best solution requiring less computation time is preferred.

3.6 Simulation

The simulation operations resemble the network analysis operations, because they also use the network topology. The network is represented by a weighted graph. The weight of an edge represents a physical property, e.g. the capacity of a road. In simulation, the time aspects and the probability model play an important role. This is a difference with network analysis. Examples of simulation are: the evacuation of an area by

road, the flow of liquids or gases through pipes, etc. [54,52].

3.7 Digital Elevation Model

There are several possibilities for the visualization of the terrain elevations. The most spectacular ones are the perspective side views with shading and hidden surfaces removed. Orthogonal projections from above are useful when height visualization techniques are used: slope hatching or shading, colour coding of the height and height contours with labels.

Besides the visualization there are other essential DEM operations, such as finding local and global extremes and line-of-sight calculations.

4 Requirements for the Data Model

As mentioned before, the backbone of a GIS will be the data model based on a spatial data structure. Some requirements for this data model are:

- *r1* Geometrical, Topological and Application data must be stored;
- *r2* It must possess good spatial properties, allowing efficient implementation of the fundamental operations described in the previous section;
- *r3* It must be possible to store the data at several levels of detail;
- *r4* It must be possible to store the data in layers of object-types, so it is possible to add or remove a layer from the display;
- *r5* The data model should support long-term storage (database aspects);
- *r6* It must be possible to add, remove or change objects and object-types.

Not all the operations in the last requirement are of equal importance. GISs are not very dynamic with regard to the geographic data. Once in a while the maps are updated by the data supplier. The creation of the initial data structure is not time critical, but is should result in a 'good' and efficient data structure. The addition of an object must be performed fast and should not decrease the performance of the data structure. Changing and deleting objects will not happen often, so these operations may be implemented less efficiently. An object may be deleted by simply setting a flag indicating that this object is not present any more. This also allows an efficient *undo*.

There should be procedures allowing easy and controlled access of the data. The data structures must be described formally, so it is possible to derive or prove properties such as the efficiency of certain operations. Because of their different nature, the three data types each require a specific data structure. These data structures will be integrated in the GIS data model.

5 Spatial Data Structures

It is hard to find a data structure which satisfies all the requirements mentioned in the previous section. Therefore, one or more requirements are temporarily dropped during the research. First we will concentrate on the spatial properties (*r2*). As mentioned in section 2 the vector format is used to store the geometrical data, that is: points, polylines and polygons. Additional structures are needed to store the relationships between these objects (topological data). A binary tree (B-tree) is inadequate to represent the spatial aspects because geometric data is multi dimensional. The same applies to other storage structures (splay trees [48]) for one dimensional data.

In the literature several solutions have been proposed for multi dimensional data and these will be briefly described: *k*-d trees, quadtrees, r-trees, Peano keys and grid files. Another data structure, which is not yet published in the context of GISs, is suggested: the BSP-tree.

The data structures described in the subsections 5.1 through 5.6 are also used in three dimensional applications. The last subsection gives some remarks on this use. Some data structures allow the geometric primitives to be stored outside the spatial data structure. Only an identifier (pointer or index) is stored in the spatial data structure and the geometric primitives are stored elsewhere. This solution is chosen in ARC/INFO [17]. Some general criteria by which the spatial data structures are compared:

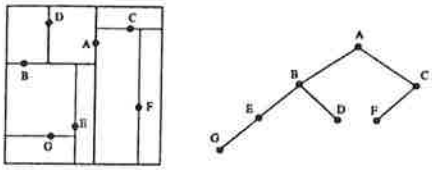

Figure 1: k-d Tree

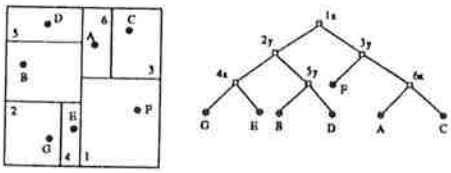

Figure 2: Adaptive$k$-d Tree

- rectangular or non-rectangular division of space;
- geometric primitives used for divisions or arbitrary regular divisions;
- simplicity of data structure and algorithms (e.g. recursive definition);
- functionality, which entities (point, line and polygon) can be stored; which operations can be performed; is three dimensional use possible?;
- efficiency, the k-d tree and the quadtree are very well known and their efficiency has been compared in [33], [42] and [5].

5.1 k-d Trees

Bentley describes this data structure in 1975 [6]. The basic form of the *k-d tree* stores k-dimensional points. Each time a point is inserted, this point is used to descend the tree until a leaf node is reached. In the two dimensional case this leaf represents a rectangle, which will be divided by the x-coordinate of the new point on the odd levels in the tree and by the y-coordinate on the even levels in the tree, see figure 1. The internal nodes contain points already inserted. The tree is descended by comparing the value of the coordinate of the point stored at the internal node with the corresponding coordinate of the search point.

A disadvantage of the k-d tree is that the shape of the tree depends on the order in which the points are inserted. In the worst case a k-d tree of n points has n levels. The *adaptive k-d tree* [7] solves this problem by choosing a splitting point (which is not an element of the set of data points), which divides the set of points into two sets of (nearly) equal size. This process is repeated until each set contains at most one point, see figure 2. The adaptive k-d tree is not dynamic: it is hard to insert or delete points.

Another variant of the k-d tree is the *bintree* [47]. The space is divided into two equal rectangles. This is repeated until each leaf contains at most 1 one point.

For practical use it can be often convenient to use leaf nodes containing more than one data point. The maximum number of points that a leaf may contain is called the *bucket* size. If the tree is not completely stored in core, the bucket size is chosen in a way that the bucket fits within one disk page.

Matsuyama et al. [33] show how the geometric primitives polyline and polygon may be incorporated using the centroids of a bounding box in the 2-d tree. Rosenberg [42] uses a 4-d tree to store a bounding box by putting the minimum and maximum point together in one 4-d point.

5.2 Quadtrees

The *quadtree* is a generic name for all kinds of trees that are built by recursive division of space into four quadrants. Samet [44] gives an excellent overview of which this subsection is a partial abstract.

The best known quadtree is the *region quadtree*, which is used to store an approximation of a polygon. First the area of interest is enclosed by a square. A square is divided into four equally sized squares until it is completely inside (a BLACK leaf) or outside (a WHITE leaf) the polygon or until the maximum depth of the tree is reached (dominant colour is assigned to leaf), see figure 3. The main drawback is that it does not contain an exact representation of the polygon. The

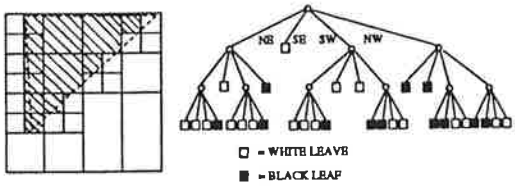

Figure 3: Region Quadtree

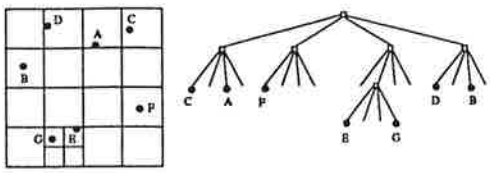

Figure 5: PR Quadtree

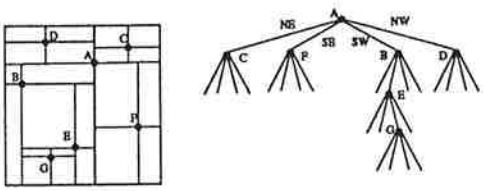

Figure 4: Point Quadtree

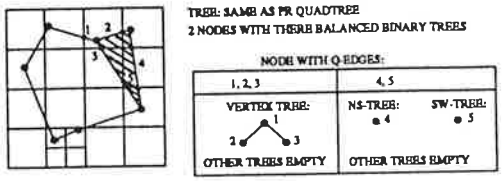

Figure 6: PM Quadtree

same applies if the region quadtree is used to store points and polylines. This kind of quadtree is useful for storing binary raster data, but, as argued in section 2, this document concentrates on vector format based data structures.

The quadtrees in the remainder of this subsection store exact (vector format) representations. The *point quadtree* resembles the *k*-d tree. The difference is that each time the space is divided into four rectangles instead of two, see figure 4.

The *PR quadtree* (Point Region) does not use the points of the data set to divide the space. Each time it divides the space, a square, into four equal sub squares, until each contains at most one point, see figure 5.

A polygonal map, a collection of polygons, can be represented by the *PM quadtree*. The vertices are stored in the tree in the same way as in the PR quadtree. The edges are segmented into q-edges which completely fall within the squares of the leaves. There are seven classes of q-edges. The first class of q-edges are those that intersect one boundary of the square and meet at a vertex within that square. The other six classes intersect two boundaries and are named after the boundaries they intersect: NW, NS, NE, EW, SW and SE. For each non-empty class, the q-edges are stored in a balanced binary tree. The first class is ordered by an angular measure and the other six classes are ordered by their intercepts along the perimeter. Figure 6 shows a polygonal map and the corresponding PM quadtree. The PM quadtree provide a reasonably efficient data structure for performing various operations: inserting an edge, point-in-polygon testing, overlaying two maps, range searching and windowing.

The remarks about buckets in the previous subsection are also valid here. The *CIF quadtree* is a kind of quadtree that is particularly suited for rectangles, a description of it can be found in [44,42].

5.3 R-trees

The *R-tree* was defined by Guttman [25] in 1984. The leaf nodes of the R-tree contain entries of the form: *(I,object-identifier)* , where *object-identifier* is a pointer to a data object and *I* is a bounding box (or minimal bounding rectangle, MBR). An advantage of the R-tree is that (pointers to) full and non-atomic

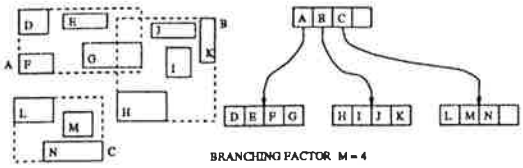

Figure 7: R-tree

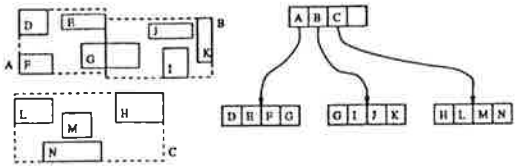

Figure 8: R^+-tree

objects (e.g. polygons) are stored in the leaf nodes, instead of lower level primitives which must be assembled to the original object. The internal nodes contain entries of the form: *(I,child-pointer)* , where *child-pointer* is a pointer to a child and I is the MBR of that child. The maximum number of entries in each node is called the *branching factor* M and is chosen to suit paging and disk I/O buffering. The INSERT and DELETE algorithms of Guttman assure that the number of entries in each node lies between m and M, where $m \geq M/2$ is the minimum number of entries per node.

Figure 7 shows a R-tree with two levels and $M = 4$. The lowest level contains tree leaf nodes and the highest level contains one node with pointers and MBRs of the leaf nodes. *Coverage* is defined as the total area of all the MBRs of the leaf nodes. *Overlap* is the total area contained within two or more leaf MBRs. In figure R-tree the coverage is $A \cup B \cup C$ and the overlap is $A \cap B$. It is clear that efficient searching demands both low coverage and overlap; a *packed* R-tree.

Roussopoulos et al. [43] describe the PACK algorithm which creates an initial R-tree that is more efficient than the R-tree created by the INSERT algorithm. The PACK algorithm frequently uses a nearest neighbour function during the creation of the R-tree. The city block distance function might be used because it is computational cheaper and it favors pairs of points or MBRs that lie along the main axes instead of diagonal orientations and this results in rectangles with less coverage.

The *R^+-tree* [18], a modification of the R-tree, avoids overlap at the expense of more nodes and multiple references to some objects, see figure 8. Analytical results indicate that R^+-trees allow more efficient searching, especially in the case of larger objects.

5.4 Peano Keys

Bitwise interleaving of the two coordinates results in a one dimensional key, called the Peano key [29]. Then a well known data structure for one dimensional storage and retrieval is used. This method is very efficient for exact match queries for points. The Peano key has the same characteristics as the linear quadtree. In a linear quadtree the leaf nodes are stored with a *locational key* and the tree structure is removed [1,44].

5.5 Grid Files

The principle of grid files is the division of the space into rectangles (grids) that can be identified by two indices, one for the x-direction and one for the y-direction. The geometric primitives are stored in these grids. In this subsection two different methods will be described. The first one is used at the land registry office in The Netherlands and will be called *Cadastral grid file*. The other one is the file structure defined by Nievergelt et al. [38].

The cadastral grid file consists of several grids, each having a different resolution. If a primitive falls entirely in a grid cell of the grid with the highest resolution then it is stored in that cell. If the primitive does not fit then the grid with second highest resolution is tried, this process is repeated until the primitive is stored, see figure 9. Each grid has a displacement, so small objects crossing a high resolution grid will fit in a lower resolution grid cell. The same technique is also used by Kleiner et al. [29]. An advantage of the cadastral grid file is that it never splits a geometric primitive. However, the method is not dynamic, some cells may contain a lot of primitives while others are empty.

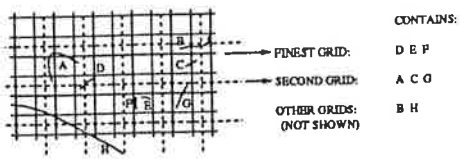

Figure 9: Cadastral Grid File

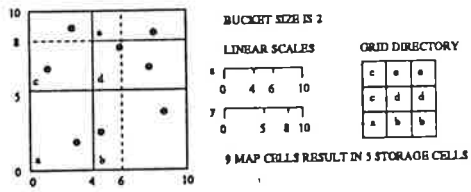

Figure 10: Grid File

The grid file as defined by Nievergelt [38] adjusts itself to the density of the data, but is also more complicated. The cell division lines need not be equidistant, for x and y there is a one dimensional array (in core) with the *linear scales*, the actual sizes of the cells. Neighbouring cells may be joined into one *bucket* if the resulting area is a rectangle. The buckets have a fixed size and are stored on a disk page. The *grid directory* is a two dimensional array, with a pointer for each cell to the correct bucket. Figure 10 shows a grid file with linear scales and grid directory. The grid file has good dynamic properties. If a bucket is too full to store a new primitive and it is used for more than one cell, then the bucket may be divided into two buckets. This is a minor operation. If the bucket is used for only one cell, then a division line is added to one of the linear scales. This is a little more complex but still a minor operation. In case of a deletion of primitives, the merging process is performed analogous to the splitting process for insertion.

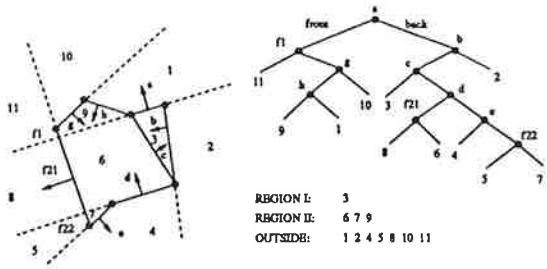

Figure 11: BSP Tree

5.6 BSP-tree

The Binary Space Partitioning (BSP) tree is a storage structure for polygons and polylines. The BSP-tree is used by Fuchs [24,23,50] to produce a hidden surface image of a static three dimensional scene. After a preprocessing phase it is possible to produce an image from any view direction in time linear in the number of polygons in the BSP-tree.

In this subsection the two dimensional BSP-tree is used for the structured storage of geometric data. It is the only spatial data structure described in this section that is not based on a rectangular division of space. It uses the line segments of the polylines and the edges of the polygons to divide the space in a recursive manner. Figure 11 shows the BSP-tree of a polygonal map. The internal nodes contain the line segments and the leaf nodes represent convex parts of the space. A disadvantage of the BSP-tree is that it is sometimes necessary to split the line segments (see edge f in figure 11) and that a polygon may be divided into two or more leaf nodes.

5.7 Three Dimensional Use

Beside the two dimensional use, nearly all solutions can be adapted for three and higher dimensional data. Some examples of applications within a GIS are:

- true three dimensional applications: storage of a Digital Elevation Model (DEM);
- storage of attributes (application data) directly with the two dimensional geometric data, resulting in 3 or more dimensions;

- time aspects add an extra dimension; for example, the change in land use.

Further research is necessary to establish if these applications are really useful, see Kraak [30].

6 Topological Structures

Topological data are not stored in the spatial data structures discussed in the previous section. Several special data structures are known for this purpose. Which topological data structure should be applied depends on how the geometric primitives are used. In one layer (see *ref* section 4) all the geometric primitives are stored in one type of topological data structure. If the geometric primitives are used to represent polygonal areas, then the TIGER structure is well suited, see subsection 6.1. The subsections 6.2 through 6.4 deal with the topological data structures in the cases of network, contour and point data.

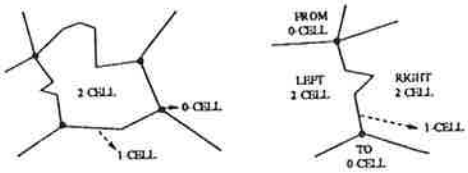

Figure 12: TIGER Structure

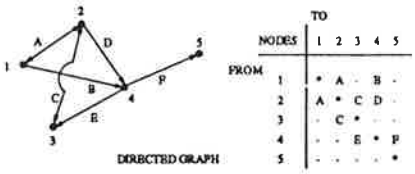

Figure 13: Network Structure

6.1 The TIGER Structure

The Topologically Integrated Geographic Encoding and Referencing (TIGER) System, is developed by the United States Bureau of the Census. The description of the TIGER structure is derived from [8].

When storing polygonal areas the structure captures the relationships between the polygon, polyline and point primitives, which are called 2-cells, 1-cells and 0-cells respectively. A 1-cell is a polyline, the first and the last points of which are by definition 0-cells. These 0-cells are called *from* and *to* 0-cells and they are the same in case of a *loop* 1-cell. The points of a 1-cell between the 0-cells are called *curvature* points and are connected by vectors. A 1-cell has two sides, a *left* and *right* 2-cell, see figure 12. The complex of 1-cells covers a finite region of the infinite projective plane. The unbounded region that surrounds the 1-cell complex is represented by a 2-cell labelled with a special code.

In addition to these definitions, the topological structure must obey two rules. The rule of *Topological Completeness* requires that the topological relationships between cells are complete, for example a 2-cell is completely surrounded by a set of 'connected' 1-cells. The rule of *Topological-Geometric Consistency* requires a consistent relationship between the geometric placement of cells and the pure topological relationships of cells, for example no two 2-cell interiors share a common coordinate.

6.2 The Network Structure

The TIGER structure can also be used for network data if the 2-cells are not used. In this way unpolygonized data with only 0-cells and 1-cells are valid. But it is not possible to represent such data as depicted in figure 13. This situation occurs in the case of a network of roads or pipelines. The arrows indicate the direction of flow. Note that there is no crossing of roads B and C. In this way a fly-over is represented.

The topological structure of this kind of data should be represented by a directed graph. The points and polylines are the nodes and edges of the graph. The polygon primitive is not used in this situation.

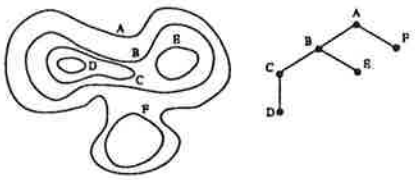

Figure 14: Contour Structure

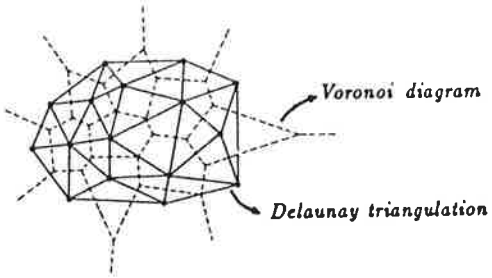

Figure 15: Voronoi Diagram

6.3 The Contour Structure

Another type of data frequently encountered in GISs is based on contours. Examples are: height contours and density iso-lines of a certain physical parameter. The nesting of the contours is described by a tree, see figure 14. A disadvantage of this tree is the unknown number of children per node.

The topological structure of contour data can be used to implement an algorithm for colouring the areas between the contours on a raster display: during the pre-order traversal of the tree, completely fill the area inside each contour that is encountered.

It is interesting to note that the Digital Feature Analysis Data (DFAD of DLMS [13]), which among others represents polygonal areas, is not using a TIGER like structure, but the contour structure as its topological base. In the case of DFAD, not the tree is stored, but the result of the inorder traversal of the tree.

6.4 The Point Structure

This last category of data seems to have no inherent topological structure. Point data can be divided into two classes: regular and irregular grid based data. Basically, point data are used for two purposes in GISs. The first one is to represent the location of objects, for example a tower. It will be clear that this will result in an irregular grid. The second type of use of point data is to represent locations at which certain parameters are measured, e.g. height of the terrain, number of people in a grid cell, air pollution. This results in regular or irregular point data.

In the case of a regular square grid, the point data degenerates into raster data. As already explained in section 2, rasters have good spatial properties and clear (implicit) topological structures. If the points are stored in a 2D-array, the neighbours of a point are found by incrementing or decrementing one of the array indices.

A useful topological structure for point data is the *Voronoi diagram* [40], see figure 15. For the points p_i for $1 \leq i \leq N$, the Voronoi diagram consists of N regions $V(i)$ with the property that if $(x, y) \in V(i)$ then p_i is the nearest neighbour of (x, y). If $H(p_i, p_j)$ is the half-plane with the set of points closer to p_i than to p_j, then

$$V(i) = \bigcap_{i \neq j} H(p_i, p_j)$$

The Voronoi diagram is used to answer spatial queries. The straight-line dual of the Voronoi diagram is called the *Delaunay triangulation* [55] and can be used in a conversion from point data to contour data.

7 A Complete GIS

A GIS is more than just a data model in which different types of data structures can be incorporated. In the following subsections important aspects like the database, the user interface and exchange standards will be treated.

There are many more interesting aspects and new developments in the GIS technology. Error models for spatial data [9,10], especially when dealing with data of multiple sources is an example hereof. Another one is the use of expert systems [41,45] to enhance the user interface of a GIS. Expert systems can also be used

during the design of cartographic maps: determining the use of colours, automatic generalization [37], placing the labels [22], etc. However, these topics do not fall within the scope of the research described in this document.

7.1 The Database

A transparent transformation from the long-term storage to the data structures in main memory should be provided. Dependent on the operator *profile* , a description of his preferences, certain functions will be performed automatically. This might include the right projection and pre-calculation and storage of frequently used properties; e.g. the area of a polygon. To store the application data (attributes), a relational DBMS can be used with references to identification codes of the geometrical entities. The database should also be used for the storage of the geometrical and topological data. The DBMS should be extendible with additional types, storage and search mechanisms, although unfortunately most databases are closed. An exception is POSTGRES [46], an experimental DBMS from the University of California (Berkeley). Some interesting design goals of POSTGRES are:

- Provide better support for complex objects instead of only integers, reals and strings.
- Provide user extendibility for data types, operators and access methods.
- Provide facilities to save and query historical data and versions.
- Make as few changes as possible to the relational model.

7.2 The User Interface

The user interface requires a careful design. The operations (section 3) are meant to be used by the operator in the first place. If he cannot work with them the GIS will not be fully used. Aspects of the user interface which deserve special attention are:

- The graphical package the be used, keeping in mind the operations described in section 3, especially the display operations. Candidates are: PostScript [3,2], CGI [27], GKS [4], GKS-3D, PHIGS [28], PHIGS+ and HIRASP [49]. Should the graphical package have 3D capabilities (for DEMs)? Should there be facilities for the transparent display of images (for overlaying polygonal areas)?

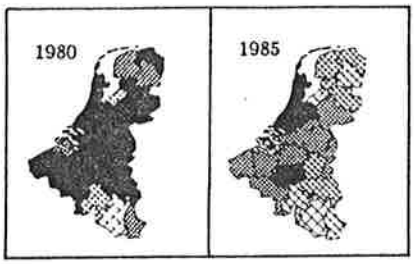

Figure 16: Multiple Maps

- Research on multi-map user interfaces, using a window based User Interface Management System (UIMS) like Windows X/11 or NeWS. Figure 16 contains some examples. Important issues are: the number of maps and other parts on the screen (e.g. for alpha-numerical information), may these parts overlap, be reshaped and moved by the operator?
- The design of a graphical query language. The selections are stated using input devices such as pick, locator, valuator and string.

7.3 Exchange Standards

The cost of data capture in GISs is usually high. An obvious solution to reduce this cost is multiple use of the same data set. Unfortunately, there are some practical problems: what data should be collected, how accurate, what format should be used? There are several exchange standards available that try to solve some of these problems. Some of them are: DLMS (DTED, DFAD) [13,12,14], USGS (DLG), SUF-2 [56], IGES [36] and GKS Metafile [26]. One of the latest efforts is 'The Proposed Standard for Digital Cartographic Data' of the Digital Cartographic Data Standards Force (DCDSTF) in which the U.S. Geological Survey plays an important role.

8 Present and Future Work

In this document the requirements for the GIS Data Model are often treated separately. The combination of requirements *r2* (spatial properties) and *r3* (several levels of detail) is expected to be quite difficult. A solution has to be found somewhere in between:

- Store everything at the most detailed level. Perform 'on the fly' generalization during the GIS operations. There are several different types of generalization: line-smoothing (some algorithms: k-th point, simple threshold and Douglas/Peucker [15,35]), join, delete, move and enlarge geographic entities. Note that the generalization process is complicated and involves human decisions.
- Try to define a discrete number of levels of detail and store them [49]. The R-tree seems to be a good spatial data structure to store multiple levels of detail. The leaf node on one level may contain a representation of the geographic entity and also a reference to a R-tree with a more detailed representation.

Further research is needed and will be conducted in this area. A prototype of the basic package for the GIS model will be implemented, using data structures described in this document. Finally, the GIS model should be tested in a non-trivial GIS application.

References

[1] D.J. Abel and J.L. Smith. A data structure and algorithm based on a linear key for a rectangle retrieval problem. *Computer Vision, Graphics and Image Processing*, 24:1–13, 1983.

[2] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley Publishing Company, 1985. ISBN 0-201-10174-2.

[3] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook.* Addison-Wesley Publishing Company, 1987. ISBN 0-201-101179-3.

[4] American National Standards Institute. Computer graphics - graphical kernel system (GKS) - functional description & Fortran binding. 1985. ANSI X3.124-1985 (includes ANSI X3.124.1-1985).

[5] D.A. Beckley, M.W. Evens, and V.K. Raman. Multikey retrieval from k-d trees and quad trees. *ACM SIGMOD (Management of Data)*, 14(4):291–301, December 1985.

[6] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[7] Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *Computing Surveys*, 11(4):397–409, December 1979.

[8] Gerard Boudriault. Topology in the TIGER file. In *AUTO-CARTO 8*, pages 258–269, 1987.

[9] P.A. Burrough. Multiple sources of spatial variation and how to deal with them. In *AUTO-CARTO 8*, pages 145–154, 1987.

[10] P.A. Burrough. *Principles of Geographical Information Systems for Land Resources Assessment. Monographs on Soil and Resources Survey No.12*, Oxford University Press, 1986. ISBN 0-19-854592-4.

[11] C.J. Date. *An Introduction to Database Systems.* Addison-Wesley Publishing Company, Massachusetts, 1981.

[12] Defence Mapping Agency. *Product Specifications for Digital Feature Analysis Data (DFAD) - Level 1 and Level 2.* Technical Report, DMA Aerospace Center, St. Louis, Missouri, April 1986. PS/ICE/200, PS/ICG/200.

[13] Defence Mapping Agency. *Product Specifications for Digital Landmass System (DLMS) Data Base.* Technical Report, DMA Aerospace Center, St. Louis, Missouri, July 1977. PS/ICD/100, PS/ICE/100, PS/ICF/100, PS/ICG/100.

[14] Defence Mapping Agency. *Product Specifications for Digital Terrain Elevation Data (DTED).* Technical Report, DMA Aerospace Center, St. Louis, Missouri, April 1986. PS/ICD/200, PS/ICF/200.

[15] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112–122, 1973.

[16] Y. Doytsher and B. Shmutter. Intersecting layers of information - a computerized solution. In *AUTO CARTO London*, pages 136–145, 1986.

[17] Environmental Systems Research Institute. Arc/info users guide. July 1987.

[18] Christos Faloutsos, Timos Sellis, and Nick Roussopoulos. Analysis of object oriented spatial access methods. *ACM SIGMOD (Management of Data)*, 16(3):426–439, December 1987.

[19] Andrew U. Frank. Overlay processing in spatial information systems. In *AUTO-CARTO 8*, pages 12–31, 1987.

[20] Wm. Randolph Franklin and Harry L. Lewis. 3D graphic display of discrete spatial data by PRISM maps. *ACM Computer Graphics*, 12(3):70–75, August 1978.

[21] Wm. Randolph Franklin and Peter Y.F. Wu. A polygon overlay system in Prolog. In *AUTO-CARTO 8*, pages 97–106, 1987.

[22] H. Freeman and J. Ahn. Autonap - an expert system for automatic map name placement. In *Proceedings 1st International Symposium on Spatial Data Handling, Zurich*, pages 544–571, 1984.

[23] Henry Fuchs, Gregory D. Abram, and Eric D. Grant. Near real-time shaded display of rigid objects. *ACM Computer Graphics*, 17(3):65–72, July 1983.

[24] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. *ACM Computer Graphics*, 14(3):124–133, July 1980.

[25] Antonin Guttman. R-trees: a dynamic index structure for spacial searching. *ACM SIGMOD 84*, 14:47–57, 1984.

[26] International Organization for Standardization. Information processing systems - computer graphics - metafile for the storage and transfer of picture description information. 1986. ISO DIS/8632/1, ISO DIS/8632/2, ISO DIS/8632/3, ISO DIS/8632/4.

[27] International Organization for Standardization. Information processing systems - computer graphics - interfacing techniques for dialogues with graphical devices - functional specification. 1986. ISO DP/9636/1, ISO DP/9636/2, ISO DP/9636/3, ISO DP/9636/4, ISO DP/9636/5, ISO DP/9636/6 (CGI).

[28] International Organization for Standardization. Information processing systems - computer graphics - programmers hierarchical interactive graphics system. June 1986. ISO C79/SC21/N819 (PHIGS).

[29] Andreas Kleiner and Kurt E. Brassel. Hierarchical grid structures for static geographic data bases. In *AUTO CARTO London*, pages 485–496, 1986.

[30] Menno J. Kraak. Computer assisted cartographic three-dimensional imaging techniques. In *AUTO CARTO London*, pages 53–58, 1986.

[31] Paul D.M.E. Lahaije. *Algorithms and data structures for efficient of CARIN roadmap data from a Compact Disc ROM*. Master's thesis, Eindhoven University of Technology, May 1986.

[32] Anthony E. Lupien, William H. Moreland, and Jack Dangermond. Network analysis in geographic information systems. *Photogrammetric Engineering and Remote Sensing*, 53(10):1417–1421, October 1987.

[33] Takashi Matsuyama, Le Viet Hao, and Makoto Nagao. A file organization for geographic information systems based on spatial proximity. *Computer Vision, Graphics and Image Processing*, 26:303–318, 1984.

[34] William H. Moreland and Anthony E. Lupien. Realistic flow analysis using a simple network model. In *AUTO-CARTO 8*, pages 122–128, 1987.

[35] J.-C. Muller. Optimum point density and compaction rates for the representation of geographic lines. In *AUTO-CARTO 8*, pages 221–230, 1987.

[36] Roger N. Nagel, Walt W. Braithwaite, and Philip R. Kennicott. *Initial Graphics Exchange Specification, IGES Version 1.0*. Technical Report, National Bureau of Standards, U.S.A., January 1980.

[37] B.G. Nickerson and H. Freeman. Development of a rule-based system for automatic map generalization. In *Proceedings 2nd International Symposium on Spatial Data Handling, Seattle*, 1986.

[38] J. Nievergelt, H. Hinterberger, and K.C. Sevcik. The grid file: an adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, 1984.

[39] F.J. Ormeling and M.J. Kraak. *Kartografie: ontwerp, productie en gebruik van kaarten.* Delftse Universitaire Pers, 1987. ISBN 90-6275-337-X.

[40] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry.* Springer-Verlag, 1985. ISBN 0-387-96131-3.

[41] William J. Ripple and Veit S. Ulshoefer. Expert systems and spatial data models for efficient geographic data handling. *Photogrammetric Engineering and Remote Sensing*, 53(10):1431–1433, October 1987.

[42] Jonathan B. Rosenberg. Geographical data structures compared: a study of data structures supporting region queries. *IEEE Transactions on Computer Aided Design*, CAD-4(1):53–67, January 1985.

[43] Nick Roussopoulos and Daniel Leifker. Direct spatial search on pictorial databases using packed r-trees. *ACM SIGMOD (Management of Data)*, 14(4):17–31, December 1985.

[44] Hanan Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260, June 1984.

[45] T. Smith, D. Peuquet, S. Menon, and P. Agarwal. KBGIS-II. a knowledge-based geographical information system. *International Journal of Geographical Information Systems*, 1(2):149–172, 1987.

[46] Michael Stonebraker and Lawrence A. Rowe. The design of POSTGRES. In *ACM SIGMOD '86*, pages 340–355, 1986. Washington D.C., May 28-30.

[47] Markku Tamminen. Comment on quad- and octtrees. *Communications of the ACM*, 27(3):248–249, March 1984.

[48] Robert E. Tarjan. Het ontwerpen van algorithmen (1986 turing award). *Informatie jaargang*, 29(9):789–796, 1987. also in: Communications of the ACM, march 1987.

[49] Wim J.M. Teunissen. *HIRASP A Hierarchical Modelling System for Raster Graphics.* PhD thesis, University of Leiden, 1988.

[50] Wim J.M. Teunissen and Peter J.M. van Oosterom. *The creation and display of arbitrary polyhedra in HIRASP.* Technical Report, University of Leiden, July 1988. Department of Computer Science, Report 88-20.

[51] J.D. Ullman. *Priciples of Database Systems.* Computer Science Press, Inc., Rockville, 1982.

[52] Ir. M.J. van de Scheur and Drs. D.J. Stolk. *Informatie- en Rekensysteem ten behoeve van de Rampbestrijding bij Ongevallen met Gevaarlijke Stoffen.* Technical Report, FEL-TNO Divisie 1, May 1986. FEL-TNO rapport no. 1986-41.

[53] J. van den Bos, M. van Naelten, and W. Teunissen. IDECAP interactive pictorial information system for demographic and environmental planning applications. *Computer Graphics Forum*, 3:91–102, 1984.

[54] Drs. P.A.B. van Schagen. Computerwargames voor training en opleiding in de koninklijke landmacht. *Militaire Spectator*, 156(3):116–122, 1987.

[55] Remco C. Veltkamp. *The γ-Neighbourhood Graph for Computational Morphology.* Technical Report, University of Leiden, June 1988. Department of Computer Science.

[56] Werkgroep SUF-2. Standaard uitwisselings formaat 2. July 1987.

PROCEEDINGS DEEL 2

CSN 88

SION

COMPUTING SCIENCE IN THE NETHERLANDS

JAARBEURS UTRECHT

3 EN 4 NOVEMBER 1988