BALTIMORE 1991

TECHNICAL PAPERS

1991 ACSM-ASPRS ANNUAL CONVENTION

Volume 6

Auto-Carto 10

1934 American Society for Photogrammetry and Remote Sensing

American Congress on Surveying and Mapping · 1941

BALTIMORE 1991

TECHNICAL PAPERS

1991 ACSM-ASPRS
ANNUAL CONVENTION

□

Volume 6

Auto-Carto 10

ACSM 51st Annual Convention
ASPRS 56th Annual Convention

ISBN 0-944426-38-7
ISBN 0-944426-44-1

Published by
American Congress on Surveying and Mapping
and
American Society for Photogrammetry and Remote Sensing
5410 Grosvenor Lane
Bethesda, MD 20814-2160

USA
Printed in the United States of America

Proceedings

Tenth International Symposium on Computer-Assisted Cartography

AUTO-CARTO 10

Baltimore, Maryland
March 25-28, 1991

FOREWORD

This volume contains the 27 papers of the Tenth International Symposium on Computer-Assisted Cartography. *AUTO-CARTO* 10 represents a new direction for the series. (Only time will tell whether it was the start of a new course, or a temporary detour!) For the first time, full papers were peer-reviewed by at least two, and often three or four, members of the program committee. Furthermore, accepted papers were returned to the authors for revision based on the reviewers comments. Our announced plan was to accept between 30 and 40 papers, and we expected to select those from a pool of perhaps 100 submissions. Based on that expectation, a large program committee with a total of 22 members was established to conduct the reviewing and to advise on final program decisions; the program committee co-Chairs wish to thank all the members of the committee for their timely and conscientious reviewing and other participation in the decision-making process.

We were surprised, and somewhat disappointed, when only 39 full papers were submitted for consideration. (This compares to some 160 *abstracts* submitted to *AUTO-CARTO* 8 four years earlier.) However, the quality of those papers was in most cases very high, and we believe that "self-selection" was in part responsible for the low number submitted. Other possible reasons for the low pool of papers are that the community realized too late that full papers were due in September; or that the community does not wish to have fully-refereed proceedings, preferring instead to submit refereed material through the journals. In fact some people suggested exactly that reason, and the program chair(s) for *AUTO-CARTO* 11 will have to consider carefully whether to try full reviewing again, or return to the previous system of screening extended abstracts or short proposals.

Despite the low number of submissions, we continued to apply the high refereeing standards that had been promised to the authors. All unanimous recommendations from the reviewers were followed; for the remaining papers, a subset of the program committee met in Anaheim and finalized the decisions. The result was the acceptance of 27 of the papers which appear in this volume. Then, because the 27 papers did not fill all available *AUTO-CARTO* 10 program slots, three panel discussions that had been proposed were inserted into program, and brief summaries (unrefereed) also appear in this volume.

Do the members of the "*AUTO-CARTO* community" want refereed proceedings? After the meeting, people should make their feelings on this clear, and transmit them to us, or to the Board of the American Cartographic Association. Regardless of the future of full-paper refereeing in *AUTO-CARTO*, we are proud of the quality of papers in this volume, and look forward to an excellent meeting in Baltimore!

David M. Mark, Buffalo, New York
Denis White, Corvallis, Oregon
28 January 1991

The *AUTO-CARTO 10* Program Committee

Table of Contents

Author Index

DATABASE ARCHITECTURE FOR MULTI-SCALE GIS

Christopher B. Jones
Department of Computer Studies
The Polytechnic Of Wales
Pontypridd
Mid Glamorgan, CF37 1DL, UK

ABSTRACT

Many applications of GIS, such as planning, exploration and monitoring of natural resources, require the mapping and analysis of spatial data at widely differing scales. Ideally, a single large scale representation of spatial data might be stored, from which smaller scale versions were derived. Currently however, automation of the necessary generalisation processes is not sufficiently well advanced for this to be a possibility. Consequently, multiple representations must be maintained, though proven generalisation techniques can be used to reduce data duplication, provided that processing overheads are not prohibitive. Maintenance of a multiple representation database requires a flexible approach to the use of both single-scale and multiresolution data structures. Furthermore, rule-based software is required for a) deciding whether new datasets should be merged with existing ones, or stored as separate representations, and b) selecting appropriate representations and applying generalisation procedures to satisfy user queries. This paper presents an overview of a database design, based on a deductive knowledge-based systems architecture, which attempts to meet these requirements.

INTRODUCTION

Increasing interest in the use of geographical information systems (GIS) brings with it requirements for the analysis and display of geographical information at various scales, relating to different locations and to different themes. The accumulation of this information introduces a need for sophisticated databases that are flexible with respect to the variety of stored data and to the scale and locational accuracy of the output. Such requirements arise in organisations concerned with monitoring or exploiting the natural and man-made environment.

Maintenance of data derived from a variety of source scales raises a major issue of whether the individual real-world objects should be represented once, at their highest resolution, or whether multiple versions at different scales should be stored. Ideally perhaps the former option appears most desirable, since it avoids data redundancy and the possibility of inconsistency between versions. The approach depends however upon the assumption that smaller scale versions can be derived automatically. With the current, relatively limited, capabilities of automatic generalisation software, this is not a valid assumption (Brassel and Weibel, 1988). In a multi-scale database servicing a wide range of output requirements there is therefore good reason to store multiple representations of the same objects (Brassel, 1985). Even when small scale versions can be derived automatically, there will be situations, involving large degrees of generalisation, in which the delays due to computation could not be tolerated in an interactive GIS. In such circumstances it could be desirable to store the results of automated generalisation.

The presence of multiple representations of spatial objects, and the need for retrieval at a range of scales, places considerable demands upon a database management system. If it is to maintain and retrieve data with the minimum of user-intervention, it must incorporate software capable of making decisions about updates and retrievals. When new datasets are loaded, decisions must be taken about whether to replace existing data, merge with existing data, or store as a separate representation. On querying the database there may be several candidate representations. One of these may be selected for output or it may be used to derive an appropriate representation using automatic generalisation procedures. In addition to the inclusion of 'intelligent' software, the need arises for data structures which are efficient for access in terms of ground resolution, spatial location, topology and aspatial attributes.

A research project has been initiated with the aim of building an experimental multi-scale spatial information system. In the remainder of the paper the components of the proposed experimental system are outlined, before discussing specific issues which arise in designing and implementing multi-scale GIS. Attention is focused in particular on multiresolution data structures, indexing mechanisms and the maintenance and query of multiple scale representations.

COMPONENTS OF A MULTI-SCALE DATABASE ARCHITECTURE

An overview of the main components of a proposed multi-scale database is illustrated in Figure 1. All updates and queries are channelled through a deductive subsystem, the rule-base of which controls changes to the contents of the database and retrievals from it. The contents of the database are summarised within an object directory which, though it may be spatially segmented, serves primarily to record the presence of stored objects in terms of their application-specific classes and the nature of their representations with regard to dimension, locational accuracy and spatial data model. The rule base of the deductive subsystem refers to the current contents of this object directory in order to make decisions about appropriate strategies for update and retrieval. It also controls the execution of spatial processors required for certain update operations and for performing, where necessary, generalisation operations on retrieved objects.

The detailed spatial structure of objects listed in the object directory is recorded in the topology and metric geometry components. The metric geometry component stores data referenced directly to locational coordinates and could include both vector and raster data employing specialised multiresolution data structures. In the case of vector structured objects a close relationship could be expected with corresponding elements in the topology component of the database. The distinction between topology and metric geometry is intended to facilitate efficient search based on topological information at various levels of detail. Range searches for all objects in a given rectangular window may be served directly by the metric geometry data structures.

The rule-based component of the experimental system is envisaged initially as a deductive, or logic database which may be implemented in a logic programming language with extensions for calling external procedures and for access to permanent storage. Execution of rules for update of single resolution and multiresolution spatial data structures and for generalisation will then be achieved by calling the various spatial

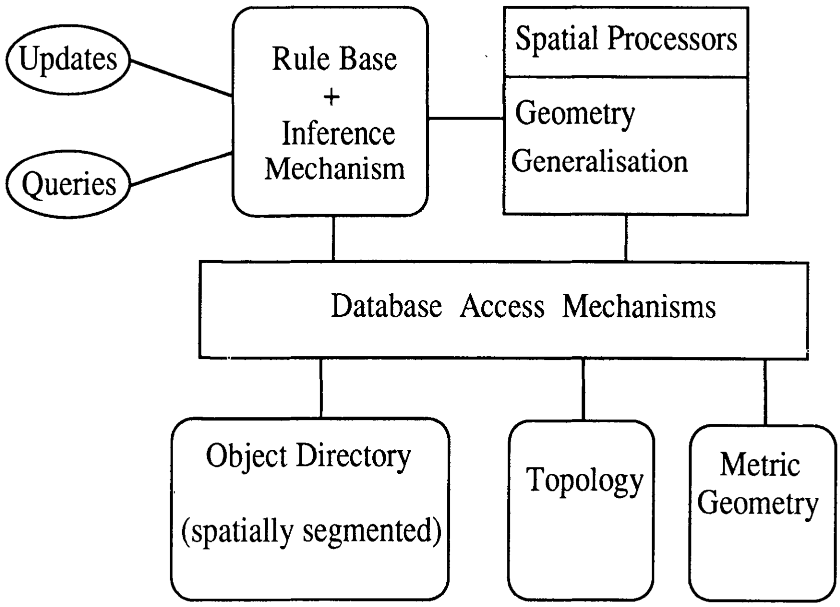


Figure 1

processors. Implementation of the spatial data structures requires the use of complex data types, while the processors which operate on them could, in some cases, consist of knowledge-based subsystems in their own right. These latter components of the database may appear therefore to be suited to implementation using object-oriented programming techniques.

Recognition of the importance of combining rule processing with object-oriented databases is reflected in the design of systems such as POSTGRES (Stonebraker, 1986). The potential of this type of database system for implementing multi-scale GIS has already been identified by Guptill (1989, 1990). From the more purely deductive database standpoint, new versions of the logic programming language Prolog are being developed to provide efficient integration with a permanent database (Bocca et al, 1989; Vieille et al, 1990). By adding facilities for handling complex data types and for calling external procedures, the deductive database architecture may also then provide a suitable basis for building multi-scale geographical databases.

MULTIRESOLUTION DATA STRUCTURES

Whether there are single or multiple representations of individual objects and classes of object, each representation may be regarded as a candidate for retrieval over a range of scales. The largest scale limit will be constrained by the locational accuracy of the geometric data. The smallest scale limit will be determined by the capability of automated generalisation functions which can operate on the object. Widely used line generalisation procedures such as the Douglas algorithm (Douglas and Peucker, 1973) have been used, in combination with smoothing operators over scale changes in excess of a factor of 100 (Abraham, 1988). When the linear features form part of areal objects, automated procedures are generally very much more restrictive, since issues such as object amalgamation and displacement must be taken into account. Automated areal generalisation was used in the ASTRA system (Leberl and Olson, 1986) but scale changes were only of a factor of about two. A variety of techniques is available for generalisation of digital terrain models (Weibel, 1987). Limits on the possible degree of generalisation of these models depends on the error tolerance of the application. However, when structure lines (ridges, valleys and form lines) are added to the model, the limits may be expected to be similar to those of the generalisation of the individual linear features.

Given that individual representations apply over a range of scales, the question arises as to how best to store the objects to achieve efficient access at different scales. The options are single storage of the object with generalisation to smaller scales at the time of the query; storage of several pre-generalised versions of the object, with the possibility of data duplication (as in Guptill, 1990); and storage of a non-duplicating hierarchical representation of the single object (see below). The first of these options could, in the case of linear features and terrain models, require initial retrieval of orders of magnitude excess data before simplification by a generalisation function. The second option could give efficient access to a representation which may closely approximate the retrieval specification, but at the expense of a storage overhead due to data duplication. The third option is a compromise in which a generalisation function is used to segregate the geometric component of the objects according to their contribution to shape and accuracy. By organising the component data in a hierarchical manner it is then possible to access only the geometric data required to build a representation at, or an approximation to, the required level of generalisation.

Multiresolution data structures which avoid or minimise data duplication are available for both linear features and surfaces. For linear features, the strip tree (Ballard, 1981) provides a means of accessing successively higher resolution approximations to a curve represented by rectangular strips. In its original form it is not very space efficient, as individual points may be stored several times if they bound successively narrower strips. Each rectangle must also be explicitly defined. The original strip tree consists essentially of a binary tree. The root node stores a rectangular strip which encloses the entire feature, along with pointers to two offspring. A point where the curve touches the side of the initial strip is used to subdivide the curve into two parts, each of which is represented by enclosing strips which are stored in the offspring nodes. The curve is divided recursively in this manner until individual strips coincide with straight line, zero width, segments between successive vertices of the feature.

The multi-scale line tree (or line generalisation tree) is related to the strip tree and may be regarded as a tree of variable branching ratio rather than a binary tree (Jones 1984, Jones and Abraham 1986,1987). Each level of the structure corresponds to a maximum implicit strip width. Furthermore, it is vertex rather than strip oriented, and each level stores vertices which are intermediate to those at the next higher level. The result is that it is significantly more space-efficient than the strip tree. It has been implemented in a network database in which each level of a hierarchy is stored independently of the other levels of the same line object, but in association with the equivalent generalisation levels of other objects (Abraham, 1988). Thus rapid access to all features of a particular resolution is facilitated by only retrieving, for each object, those hierarchical levels which are relevant to a specified output scale (or spatial resolution).

Use of a multi-scale line tree introduces the problem of maintaining aspatial and topological attributes of the line features. If the hierarchy extends across a wide range of scales, the line itself may be geographically extensive such that there are distinct internal subdivisions relating to different feature codes and to topological nodes. By attaching sequence numbers to the component vertices of a line, aspatial classification and topological structure can be defined in terms of ranges of sequence numbers and individual sequence numbers which have been designated as nodes (Jones and Abraham, 1987; Abraham, 1988). To retain access efficiency, the node vertices should be stored at the highest hierarchical level (lowest resolution) at which they are likely to be required. Thus vertices which the generalisation procedure may classify as low level would, if they were logical nodes, be raised to the appropriate higher level.

Multiresolution representations of surfaces may be categorised into those based on mathematical functional models of the surface and those based on original, or derived, sample points. If the coefficients of a surface function are orthogonal, in the sense that they represent independent components of the surface shape, then a multiresolution data structure could be created by separating the storage of the components into distinct records. Each record would correspond to a 'level', characterised by the extent to which the stored coefficients contributed to the surface shape. The most important components could be stored at the highest levels, while less significant ones were stored at progressively lower levels. A problem which occurs when using global functions, such as Fourier Series, is that the reconstructed, simplified surface, may be subject locally to relatively large errors. A mathematical function approach which controls errors can be obtained by partitioning the surface into rectangular regions each of which is represented by its own function (Pfaltz, 1975). By partitioning the surface in the manner of a quadtree, the size of the quadrants can be reduced locally until the chosen function fits the surface to within a pre-specified tolerance (Chen and Tobler, 1986; Leifer and Mark, 1987). Although the method has been applied primarily to the representation of surfaces at a specified error tolerance, it could be extended into a multiresolution quadtree in which intermediate (subdivided) nodes stored a function accompanied by a measure of the associated error.

Surfaces represented by sample points are usually organised either as regular grids of elevation values or as an irregular set of significant points. Irregularly distributed points are typically structured by triangulation, to form a triangulated irregular network, or TIN (Peucker et al, 1978). Because the sample density of a TIN is adapted to local variation in surface detail the structure lends itself to implementation as a multiresolution structure.

The Delaunay pyramid (De Floriani, 1989) is a hierarchical multiresolution tree for storing triangulations. The top level of the tree stores a Delaunay triangulation of a subset of the original dataset of important points. The next lower level is constructed by adding vertices which are chosen to be the most distant from the previous triangulated surface. Points are added to the previous surface, which is re-triangulated to accommodate them until the error between this new surface and the remaining points is within a pre-set tolerance. The next lower level is created in a similar manner, controlled by the error tolerance for that level. Each level stores a list of the triangles and vertices of which it is composed, the differences (in terms of triangles) between the adjacent upper and lower levels, and pointers from certain triangles to those which replace them, and are hence intersected by them, at the immediate lower level. Note that only a subset of triangles at each level points to lower triangles, since some of the previous triangles will be retained in the lower level.

An advantage of a triangulated surface model is that it provides the possibility of being integrated with point, linear and polygonal features. If the vertices which define the latter features are merged with those which define a digital elevation model then, after triangulation, the linear and polygonal features can be constituted by the edges within the triangulation, while point features are represented by single nodes. To ensure that linear features are retained in this way, the triangulation process must be constrained by boundaries defined by the linear features (see De Floriani and Puppo, 1988, for the constrained triangulation of multiresolution topographic surfaces). Provided all nodes are uniquely identified, the embedded spatial objects and their topology can be referenced directly to sequences of, and individual, triangulation nodes. In a multiresolution structure, references to nodes can include their level within the hierarchy and, just as with the multi-scale line tree topology, their nodes would be stored at the highest level that they could be expected to be of use. A multiresolution triangulation data structure integrated with topology and feature specification is currently being developed (details will be published elsewhere).

INDEXING MECHANISMS

Appropriate schemes for efficient spatial access to multiresolution hierarchies may vary according to whether the objects encoded in the hierarchies are very extensive compared with potential regions of interest. This factor determines the desirability of incorporating spatial indexing within the object representation in addition to a spatial index which refers only to the entire objects. The latter indexing scheme would indicate the storage location of objects, the geometry of which was stored in, for example, a multi-scale line tree, a multiresolution triangulation or a single level representation. Methods of implementing the primary object index include techniques such as i) a fixed grid with references to intersecting objects; ii) a bounding quadtree cell scheme (Abel and Smith, 1983); and iii) an R-tree, or one of its relatives, which works with minimum bounding rectangles (Guttman 1984, Faloutsos, 1987). Depending on the nature of the application, an additional aspatial index to objects could also be desirable.

If the geometry of objects referenced by the spatial or aspatial index was extensive compared with the search window, it would be necessary to traverse the geometric data structure, selecting those parts inside the window. If the geometry was stored as a multiresolution hierarchy (line tree or triangulation), covering a wide range of scales, then it could frequently be expected to be spatially extensive relative to query windows for large scale applications. The multi-scale line tree was implemented on this assumption and incorporated spatial indexing within each level of the hierarchy. In that experimental database, both fixed grid and quadtree schemes were applied, in which the cells of the grids and of the quadtrees stored sets of vertices in chained records. When the fixed grid size was selected to be different for each level (according to a regular pyramid) the performance of the two schemes was found to be similar (Abraham, 1988).

In De Floriani's Delaunay pyramid (De Floriani, 1989), the pointer-based implementation provides some direction to spatial search within the structure once candidate triangles have been identified at the top level. The implementation described appears to have been oriented towards point rather than window searches. An alternative approach, currently being pursued, is to impose a spatial index on each level.

Bearing in mind that a multi-scale database may be very large and that objects may occur at widely differing levels of class-generalisation hierarchies, the concept of a single spatial index and a single list or index of objects becomes rather monolithic. Given that the scale of the output can be expected to be correlated with the level of class generalisation, a natural development of the indexing system is to segregate it into generalisation levels allowing direct access into an appropriate level. Each level could be associated with some limiting spatial resolution and would reference only classes of object which were regarded as likely to become significant at that scale. The choice of classes could be somewhat arbitrary on the assumption that a data dictionary indicated the correlation between class and level. It would not be necessary to refer explicitly to the parents of classes in a class-generalisation hierarchy, provided the content and structure of all such hierarchies was stored separately, allowing them to be inferred (see, for example, Egenhofer and Frank, 1989).

MAINTENANCE OF MULTIPLE REPRESENTATIONS

An important issue in maintaining multiple representations is the extent to which data duplication and data redundancy are to be tolerated. Duplication will occur when one representation is a simplified version of the other if its geometry, such as the vertices of a line or triangulated surface, is a subset of that of the other version. If an automatic procedure exists for performing the simplification, then the smaller scale version may be regarded as redundant. Data redundancy in this sense can also arise in the absence of data duplication provided that there is an automatic procedure for deriving a required small scale version from the larger scale version. For the purposes of an interactive information system however, this notion of redundancy may be questioned if the processing required by the automatic procedure was too much to provide an acceptable response time.

The multiresolution data structures referred to earlier give rapid access to generalised versions which are geometric subsets, and they therefore provide a means of avoiding data duplication, at least for linear features and surfaces. When 'quantum leap' differences occur in the course of generalisation, due for example to changes in dimensionality and to merging and displacement of objects, the existing types of multiresolution data structures cannot be used. It can also be expected that where automatic procedures do exist for this degree of generalisation, there is a greater chance of being too slow for satisfactory user interaction. It is in the event of major changes in the geometric representation that the storage of multiple versions is most likely to be appropriate. This does not however preclude the use of multiresolution data structures for separately maintaining both the smaller and larger scale representations across their different ranges of scales.

Another situation in which multiple versions might be stored is that in which data duplication was very localised, due to the presence of geographically small areas of large scale, high resolution data within a region which was covered by a much more extensive, smaller scale representation. A method of maintaining a consistent representation at the small scale, while also avoiding the data redundancy, would be to generate a multiresolution data structure from the large scale data and merge it, at the top level, with the existing small scale version. This would involve cutting out the duplicated section and edge matching between the two versions (see Monmonier, 1989b, for a discussion of techniques for automatic matching of map features which differ in their original scale of representation). It may be envisaged that the processing overheads incurred in local deletions followed by merging of the new data may not be deemed justifiable for relatively small quantities of data, since the coverage at the larger scales would only be patchy. As more extensive coverage at the larger scales accumulated in the database, a point would be reached at which the delete and merge process became justifiable.

Control over the decision on when to merge new data with stored data can be placed within a rule base which is integral to the database management system. An analogy may be made with trigger mechanisms which have been incorporated in database systems such as POSTGRES (Stonebraker, 1986). Triggers are an automatic means of maintaining integrity based on rules which dictate that once a particular data element has changed, it may propagate a sequence of changes to related records in the database. Each trigger may be expressed as a production rule which is implemented by a forward chaining mechanism in which the firing of one trigger may lead to subsequent firing of another trigger.

The possibility of a chain of triggered updates can be envisaged in a multiple representation database if the insertion of large scale representations filled gaps in an intermediate scale representation, enabling the latter to be merged with an existing, smaller scale, representation. Thus databases which include trigger mechanisms can be seen, to some extent, as dynamic, self-maintaining systems. If there was any doubt about the reliability of such systems, with regard for example to correct matching and merging of geometry and topology, these updates could be subject to user-verification before being committed to the database. All operations could be reversible if historical records were maintained in archival memory.

DATABASE QUERIES ON MULTIPLE REPRESENTATIONS

A query to a multi-scale, multiple representation database can be expected to be faced with a choice of versions which are candidates for retrieval. An automatic query processor would then need to make a choice of the appropriate retrieval to meet the user's requirements. Criteria for an appropriate retrieval would differ according to whether the output was required for analytical purposes or solely cartographic purposes. In the latter case the version retrieved might be the one which most closely resembled the level of generalisation dictated by the map's theme and scale. Such a version could be obtained by a variety of means. There could be a single level stored representation of the appropriate generalisation. Alternatively there could be a multiresolution data structure which encompassed the required generalisation level and could therefore be traversed to construct the output. Failing that, there could be a large scale version which could be generalised by software. In the latter case the automated generalisation process could operate only on that large scale version or perhaps, as Monmonier (1989a) has proposed, an additional smaller scale version could be used to guide generalisation to an intermediate level. If no sufficiently large scale data were available, a poorer quality version could be retrieved and the user warned accordingly, or a failure reported.

The above strategies would not in general be suitable for queries based on the need for data analysis problems in which locational accuracy was of prime importance. Cartographic generalisation would not then be desirable and the appropriate version would be that derived directly from, or a subset of, the largest scale representation. Particular problems could arise with this sort of query if coverage of the query window required access to representations with differing locational accuracy. In any event, data retrieved for analytical purposes would need to be labelled with their accuracy, and processes involving overlay between different objects would need to maintain a measure of the errors propagated by the combination of geometric objects.

It is apparent that implementation of a query processor capable of adapting to user requirements will require the specification of rules to control the action to be taken under the various conditions of user needs and data availablility. The query processor could operate initially on the object directory which recorded the class, location, dimension, accuracy and spatial data model of objects stored in the data base. The rules could then be applied to select the best representation given the query conditions. This would include taking the decision on whether to apply automatic generalisation procedures and choosing which procedures were most suitable. The mechanism for implementing a deductive system governing queries may differ somewhat from that governing updates, referred to in the previous section. Because a query may be regarded as a specific goal, it lends itself to a backward chaining mechanism which attempts to match the contents of the database with the search conditions.

SUMMARY

The construction of a database, capable of maintaining multiple scale representations of spatial objects, poses major problems with regard both to the development of efficient multiresolution data structures and to controlling update and answering queries. The need for explicit rules governing update, database integrity and the retrieval of generalised objects indicates the desirability of a deductive, knowledge-based architecture providing declarative rule specification. Storage of complex objects in specialised data structures, along with the need for associated processors for update and generalisation, suggests however that it may also be appropriate to use object-oriented programming techniques. A research project is currently in progress with the aim of experimenting with deductive databases for implementing a multi-scale spatial information system. In the planned system, rules of update and query processing are specified in a deductive, logic database which is interfaced to spatial processors and spatial data structures which may be implemented, at least in part, in procedural or object-oriented languages. The operation of the spatial processors may themselves employ knowledge-based inference techniques which are encapsulated within the respective modules. The primary, deductive component of the system makes decisions about appropriate update and retrieval operations by referring to the current contents of an object directory, which summarises the nature of stored object representations in terms of their feature class, location, dimension, accuracy, and spatial data model. Details of the spatial structure of stored objects are maintained within separate topology and metric geometry components of the database, to which the object directory refers.

REFERENCES

Abel, D.J. and J.L. Smith 1983, A data structure and algorithm based on a linear key for rectangular retrieval: Computer Vision, Graphics and Image Processing, Vol. 24, pp. 1-13.

Abraham, I.M. 1988, Automated Cartographic Line Generalisation and Scale-Independent Databases, PhD Thesis, The Polytechnic of Wales.

Ballard, D.H. 1981, Strip trees: a hierarchical representation for curves: Communications of the ACM, 24, pp. 310-321.

Bocca, J., M. Dahmen, M. Freeston, G. Macartney, P.J. Pearson 1989, KB-PROLOG, a PROLOG for very large kowledge bases: Proceedings 7th British National Conference on Databases, Edinburgh, pp. 163-184.

Brassel, K.E. 1985, Strategies and data models for computer-aided generalization: International Yearbook of Cartography, Vol. 25, pp. 11-28.

Brassel, K.E. and R. Weibel 1988, A review and conceptual framework of automated map generalization: International Journal of Geographical Information Systems, Vol. 2, No. 3, pp.229-244.

Chen, Z.-T., and W. Tobler 1986, Quadtree representations of digital terrain: Proceedings Auto Carto London, Vol. 1, pp. 475-484.

De Floriani, L. 1989, A pyramidal data structure for triangle-based surface description: IEEE computer Graphics and Applications, March 1989, pp. 67-78.

De Floriani, L. and E. Puppo 1988, Constrained Delaunay triangulation for multiresolution surface description: Proceedings Ninth IEEE International Conference on Pattern Recognition, CS Press, Los Alamitos, California, pp. 566-569.

Douglas, D.H. and T.K. Peucker 1973, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature: Canadian Cartographer, Vol. 10, No. 2, pp.112-122.

Egenhofer, M.J. and A.U. Frank 1989, Object-oriented modeling in GIS: inheritance and propagation: Proceedings Auto-Carto 9, Ninth International Conference on Computer-Assisted Cartography, Baltimore, Maryland, pp.588-598.

Faloutsos, C., T. Sellis, N. Roussopoulos 1987, Analysis of object-oriented spatial access methods: Proceedings ACM SIGMOD'87, pp. 426-439.

Guptill, S.C. 1989, Speculations on seamless, scaleless cartographic data bases: Proceedings Auto Carto 9, Ninth International Conference on Computer-Assisted Cartography, Baltimore, Maryland, pp. 436-443.

Guptill, S.C. 1990, Multiple representations of geographic entities through space and time: Proceedings 4th International Symposium on Spatial Data Handling, Zurich, pp. 859-868

Guttman, A. 1984, R-trees: a dynamic index structure for spatial searching: Proceedings ACM SIGMOD'84, pp.47-57.

Jones, C.B. 1984, A tree data structure for cartographic line generalisation: Proceedings Eurocarto III, Research Center Joanneum, Institute for Image Processing and Computer Graphics, Graz.

Jones, C.B. and I.M. Abraham 1986, Design considerations for a scale-independent database: Proceedings, Second International Symposium on Spatial Data Handling, Seattle, pp.384-398.

Jones, C.B. and I.M. Abraham 1987, Line generalisation in a global cartographic database: Cartographica, Vol. 24, No. 3, pp.32-45.

Leberl, F.W. and D. Olson 1986, ASTRA - A system for automated scale transition: Photogrammetric Engineering and Remote Sensing, Vol. 52, No. 2, pp. 251-258.

Leifer, L.A. and D.M. Mark 1987, Recursive approximation of topographic data using quadtrees and orthogonal polynomials: Proceedings Auto-Carto 8, Eight International Conference on Computer-Assisted Cartography, Baltimore, Maryland, pp. 650-659.

Monmonier, M. 1989a, Interpolated generalisation: cartographic theory for expert-guided feature displacement: Cartographica, Vol. 26, No. 1, pp. 43-64.

Monmonier, M. 1989b, Regionalizing and matching features for interpolated displacement in the automated generalisation of digital cartographic databases: Cartographica, Vol. 26, No. 2, pp.21-39.

Peucker, T.K., R.F. Fowler, J.J. Little, D.M. Mark 1978, The triangulated irregular network: Proceedings Digital Terrain Models (DTM) Symposium, ASP-ACSM, St. Louis, pp. 516-540.

Pfaltz, J.L. 1975, Representation of geographic surfaces within a computer: in Display and Analysis of Spatial Data, Edited by J.C. Davis and M. J. McCullagh, Wiley, pp. 210-230.

Stonebraker, M.R. and L.A. Rowe 1986, The design of POSTGRES: Proceedings ACM SIGMOD'86, pp.340-355.

Vieille, L., P. Bayer, V. Kuchenhoff, A. Lefebvre 1990, EKS-V1, a short overview: Proceedings AAAI-90 Workshop on Knowledge Base Management Systems, Boston.

Weibel, R. 1987, An adaptive methodology for automated relief generalization, Proceedings Auto-Carto 8, Eight International Conference on Computer-Assisted Cartography, Baltimore, Maryland, pp. 42-49.

RESOLUTION REVISITED

Ferenc Csillag
Department of Geography, Syracuse University
Syracuse, NY 13244
(FCSILLAG@SUNRISE.bitnet)

ABSTRACT

This paper discusses the nature of models applied primarily for environmental data, where, theoretically, data collection is not restricted in terms of resolution. Once these data are entered into a geographical information system, its data structure should also be adjusted to the underlying model. This adjustment can determine a range of scales for spatial primitives to be efficiently handled by the system. The paradox of data models, in terms of what is an object rather than a group of points, is shown with an example. It is concluded that there may not be a generally best resolution for a given environmental variable to be mapped.

INTRODUCTION

Resolution, as such, would be most frequently defined in dictionaries as technical limitation or characteristics of some kind of a system. Obviously it is associated with "the minimum difference between two independently measured or computed values which can be distinguished by measurement or analytical methods" (NCDCDS, 1988). Concerning a geographical information system (GIS), this definition would determine our task: Target objects to be mapped should be defined so that they can be distinguished from each other. This formal requirement would subsequently determine the amount of necessary detail to represent these objects. With abstract spatial entities defined, attribute properties can easily be assigned to them: census tracts have a population, square meters do not. In most instances of environmental mapping, however, the problem is faced from a different angle. First, the spatial entities should be defined according to which attributes can be assigned, since, for example, a census tract may not have high suitability for wheat. Secondly, the definition above treats distinguishability as a dichotomous variable and does not specify levels of accuracy. This is primarily due to the still existing gap in understanding the relationship between spatial and non-spatial resolution (see Dueker, 1979 for early reference) that can be referred to respectively as a recognition/identification problem in the mapping space and in the feature (or measurement) space (see Fig.1).

Let us treat the above outlined apparent contradiction in a "historical" context, i.e. with the analysis of the considerably long history of philosophical and sophisticated discussions in the GIS-era about the relationship of geometry and attributes, as well as their respective accuracies. There are numerous approaches to such issues from geosciences, cartography, statistics, etc., but unfortunately the more authoritative definitions read, the more confusing they are.

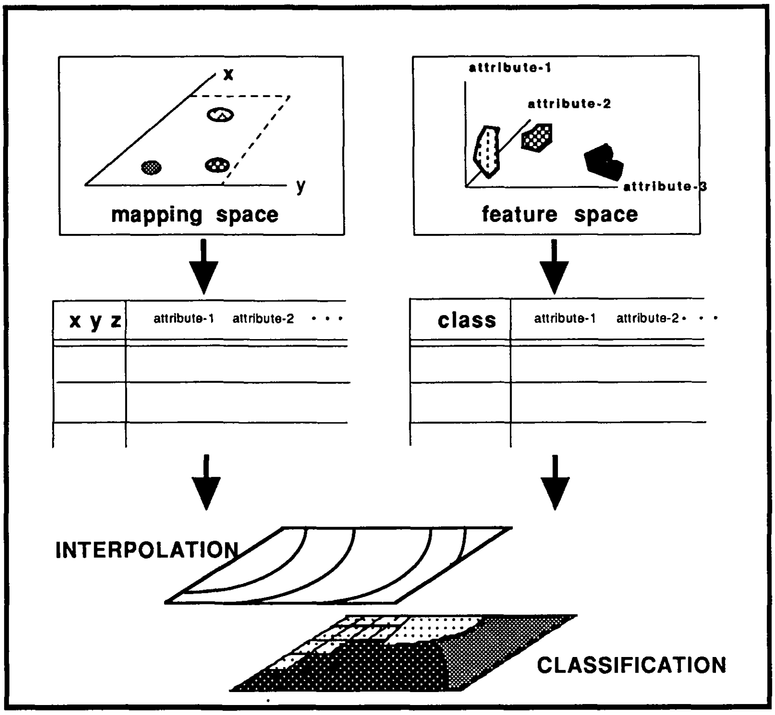


Figure 1.
Schematic representation of the relationship
between spatial and non-spatial data characteristics

The examination of the problem is organized as follows. Section 1 describes the major distinct approaches, which deserve much attention. I would argue that, although some of the technical ideas have been around for two decades or more, their authors might have wanted to use and interpret them in an inadequate way. Therefore a significant section is devoted to the mathematical models and some examples are elaborated on to prove their use. Section 2 then introduces an uncertainty relationship between spatial resolution and attribute accuracy. It is an extension of the "control one, measure another" scheme (Sinton, 1978), because it shows ***how*** resolution will vary in the mapping space once attribute accuracy is fixed, and vice versa. Section 3 presents an illustrative ecological site characterization example.

APPROACHES TO RESOLUTION

The nature of the approaches to resolution issues in mapping varies because the primary task is considered to be different: (1) in the geosciences it is assumed to be based on stochastic signal reconstruction which is also the most popular view of those in remote sensing and image processing, (2) in "conventional" cartography it is more or less loosely linked to scale and observable detail, while (3) in the jargon of digital cartography ("data modelers") representation and model-fitting are the preferred key terms. For this discussion let us use an "ultimate" working definition of our task: derive information, or in other words, make a prediction at a "non-visited" site, where site refers to both mapping and feature spaces. Additionally, when discussing these approaches, one should not forget that all our mathematical tools operate on the foundations of mathematical models, i.e. much of our effort is focused on constructing meaningful models and the sometimes lengthy demonstration of mathematical apparatus must not hide this significant first step.

Geosciences - sampling, interpolation and variability

There is an obvious assumption about objects, or processes in space, namely, the larger the sample we have, the better. Since usually a number of constraints (e.g. time, storage, money) limit our ability to sample "infinitely", models, predicting our information loss with sampling, are of extreme interest. Therefore, not surprisingly, following paths of the "digital revolution of the 50's" in geosciences (see Clearbout, 1976, Webster, 1977 for reviews), references to the ***sampling theorem*** have emerged in the general cartographic literature (e.g. Tobler, 1969, Csillag, 1987, Tobler, 1988). There are three very attractive aspects to this approach: (1) it can be utilized in sampling design, (2) it provides handy tools for interpolation as well as filtering, and (3) it is computationally very efficient.

Once one adopts this approach, the underlying mathematical-statistical assumptions of the model should be clearly understood. A significant part of the discussion below is written in order to outline the background of the choices one can have when applying mathematical models. It turns out, that in some cases certain assumptions are made not because they provide more reasonable basis, but because of the practical reason that otherwise certain problems could not have been handled. First of all, in this particular case, having a sample of size **n**, the model is concerned with $\xi(x_1),\ldots,\xi(x_n)$ stochastic variables having joint normal distribution. It is crucial to everyday practice that we hardly have any tools to check this assumption. It is especially difficult, because the sample taken at **n** locations is a single realization of the variables. Furthermore, it is assumed that the expected value of this distribution is zero, and the variance is finite. So with this model we are confined in our prediction to the case, when, somehow, our original problem has been reduced to a zero-mean variable. With these assumptions we can prove that the covariance exists (i.e. $\mathrm{COV}[\xi(x_i),\xi(x_j)]<\infty$) and it is positive semi-definite. It is our task now to construct an estimate of our distribution so the variance of the difference between the model and the estimate should be minimum. It is only due to the joint-normality assumption that our search for the estimate can be restricted for linear functions, i.e. in the form of weighted sum:

(1) $$\hat{\xi}(x) = \sum_i \lambda_i \xi x_{(i)}$$

The major problem in constructing our estimate is that we may not have sufficient information about the covariance, therefore further assumptions will be necessary. For instance, stationarity is a quite frequent assumption in order to reduce dramatically the number of elements to be estimated in the covariance matrix.

It is probably the advantage of modeling with linear functions that makes interpolation and filtering so popular in applying these tools (for math-intensive review of spectral analysis see Bracewell, 1965, or Bendat and Piersol, 1986). However, even if our assumptions are valid, there are lmany manners of abuse. When I say abuse, I mean that you can rarely find anyone who would apply these techniques, usually available by pressing a button, having tested accuracy constraints.

Let us just consider two simple cases for demonstration, linear interpolation and moving averaging. For the former case, suppose that we have taken sparse samples. Disregarding the distortion that may be due to undersampling, (i.e. less frequent sampling than half of the shortest wavelength represented), let us linearly interpolate among our data points! The total RMS-error (Bendat and Piersol, 1986), the square root of the mean difference between the original and the interpolated signal over the entire Nyquist-interval, will be

(2) $$E_{RMS}(f') = 2 - \mathrm{sinc}^2(f')(2+(2(\P f')^2/3))$$

where sinc denotes the sine-cardinal function [$\mathrm{sinc}(a)=\sin(\P a)/\P a$], while f′ denotes dimensionless frequency (equals frequency times sampling distance). As Figure 2 clearly illustrates, linear interpolation can severely distort higher frequency signals. If, for example, one would interpolate 1,2,4,... points between existing data points, the maximum error term (from Eq.2. at f′=0.5) would be -5.63, -26.83, -50.21 in decibels, and 52, 4, 0.3 in relative percentage, respectively, providing upper limits for accuracy.

Considering moving averaging, it is again the frequency-dependent distortion that should be pointed out. In general, filtering can be written in the form

(3) $$y_k = \sum_i c_i x_{k+i} \qquad (i = -N,N)$$

for which moving averaging is a special case with $c_i = 1/(2N+1)$ for all i's. The amplitude response (or frequency modulation function, S(f′)) can be obtained with the Fourier-transform of the (filter) coefficients. In this particular case it is in the form of a geometric series:

(4) $$S(f') = \sum_i c_i \exp\{-j2\P if'\} = 1/(2N+1) \sum_i \exp\{-j2\P if'\} =$$

$$= 1/(2N+1) \exp\{-j2\P Nf'\} [1 + \exp\{-j2\P f'\} + \dots + \exp\{-j2\P 2Nf'\} =$$

$$= \mathrm{sinc}(2N+1)f'/\mathrm{sinc}(f')$$

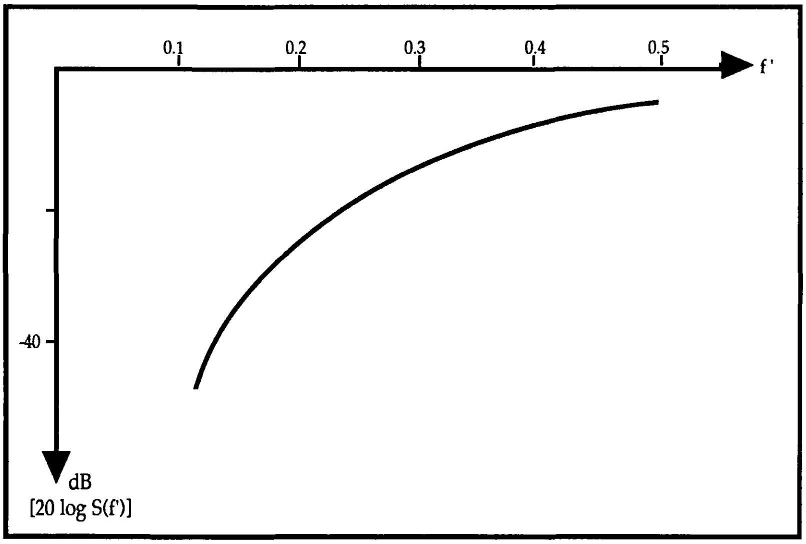


Figure 2.
Total RMS-error of linear interpolation for the Nyquist-interval
[The vertical axis is given in dB:=20logS(f')]

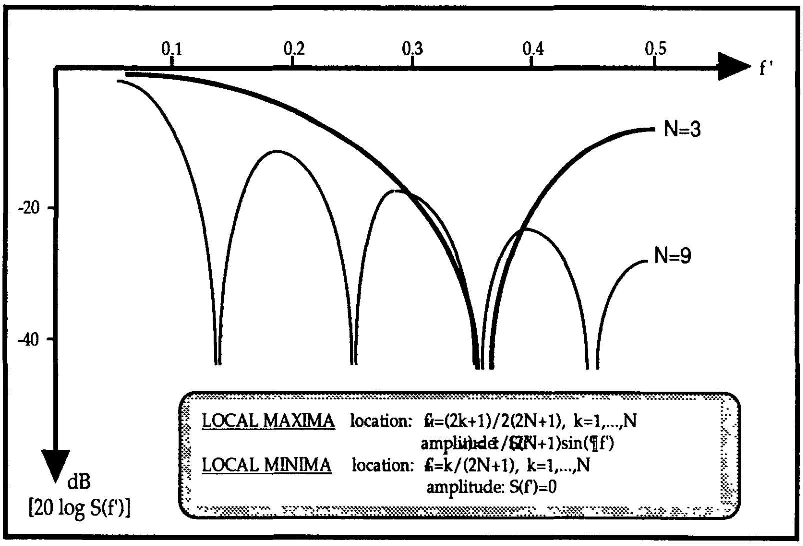


Figure 3.
The amplitude modulation transfer function of moving averaging for filter-size 3 and 9. Local extremes can be calculated with the given formulae, based on S(f'), where N denotes the length of the filter.

Some characteristics of S(f') for "everyday-size" moving average filters are displayed on Figure 3. It should be noted again, that these filters, generally applied heuristically, are close to our expectations at low frequencies, but at higher ones they seem to misbehave.

Such methods of spectral analysis aim to construct our estimate of the covariance matrix based on the strict stationarity assumption. A close relative, called kriging, became popular and uses the assumption of second order stationarity (Journel and Huijbregts 1978). The estimation procedure, in this case, is even further reduced, since it aims at the most commonly independent, parametric estimation of the substitute of the covariance function, called a semi-variogram (McBratney and Webster 1986). We should point out that it is the equivalence of the squared deviation from the mean and the normalized square difference between all pairs, known since the early days of mechanics, that is behind this methodology.

There are some further necessary remarks to be made about kriging. The estimation procedure with the stationarity assumption already eliminated a number of unknown parameters, and the covariance became a function of distance. Thus the covariance matrix is only dependent on the spatial arrangement of the sample that is, again, computationally efficient. However, the estimation procedure becomes highly dependent on the values of the semi-variogram at small distances, i.e. the nugget value, (Ripley 1981), and becomes statistically unstable when this value is not zero (Mardia, 1980, Philip and Watson, 1986). Still, the popularity of kriging is due to its close links to spatial variation (variability, heterogeneity, etc.) and the seemingly straightforward manner in which it treats continuous functions characterizing such, otherwise hardly mappable, phenomena.

The spread of these methods in GIS-applications can probably be attributed to their ability to give direct estimates of deviation from an expected value for points, as well as for areas (Journel, 1986). The spatial mean derived this way for arbitrary spatial partitioning has been widely applied in environmental sciences as well as in remote sensing (Burgess and Webster, 1980, Woodcock and Strahler, 1984). This implies that our software eventually can map not only a certain variable, but its reliability.

Cartography - scale, precision and detail

My impression is that cartographers do not like the term resolution (Robinson et al., 1984, Campbell, 1991). Implicitly, however, a kind of a rule of thumb is used according to Tobler (1988): Since the smallest physical mark which the cartographer can make is about one half of a millimeter in size, one can get a fairly good estimate of resolution in meters by dividing the denominator of map scale by two thousand.

This rule is certainly far from being absolute. The real art in cartography is to represent objects even if they are smaller than this nominal resolution because of "relative importance". Discussions about generalization, in fact, clearly reflect this paradox. For example:

> "Cartographically speaking, it is essential to retain both the details required for geographical accuracy and required for recognizability within a digital data base.
>
> ...
>
> To preserve accuracy and recognizability automatically during map generalization, one must be able to describe digitally the details that must be preserved." (Buttenfield, 1989)

Inevitably, cartographers, in the "traditional" sense (Vasiliev et al., 1990), are concerned with visually conceivable objects, i.e. map elements whose geometric and attribute characteristics are merged forming a graphic attribute. Thus the distinct boundary between precision and accuracy seems to be intentionally loosened.

In the previous section, for example, precision could have been understood as the definite upper limit of accuracy in both mapping and feature spaces, while here it is related only to location, and the content has been switched to recognizability. Consequently, this approach forms a counterpart of the one discussed above with extreme "geometrization" of the resolution issue.

Data model(er)s and structures - raster vs. vector

In one of the most recent summaries on accuracy-related research in GIS (Goodchild and Gopal, 1989), resolution had a roughly equal number of references (18) in the index with filtering (9) and interpolation (10), and generalization, on its own (17), was very close. This may mislead us into thinking of a delicately balanced approach.

The conventional separation of spatial data into geometry and attributes has not left this community yet. Such a separation is consistent with an entity-relationship model of phenomena, with geometry defining the objects, which then have attributes and relationships (Mark and Csillag, 1989). And there seems to be a borderline: Those who go for the priority of geometry (mapping space), having their roots in e.g. cartography or surveying, take a model of space most commonly called "vector", while those who emphasize the significance of classification, most probably rooted in geosciences, would adopt a model usually called "raster". Geometry and attributes, however, have in many cases intrinsic links to each other, therefore any treatment of one in isolation from the other will have a high risk of misrepresenting the phenomenon.

There is also a substantial difference between accuracy concepts in the vector and raster models (Chrisman, 1989, Mark and Csillag, 1989). The former, modeling space occupied by objects, attaches accuracy measures to representation of geometry (mapping space), while the latter, partitioning space into units which then will have attributes, prefers to assign such measures to the classification of attributes (feature space).

Regarding previous comments on the philosophy of modeling, once we have adopted a model, there are no mathematical-statistical tools to exchange it for another model; one can either apply it successfully with proper predictions, or can fail to get close to reality. In light of this, there is no valid conclusion available to decide which model is "better".

As far as choice or design of data structures is concerned, lots of efforts have been devoted to handling numerous kinds of objects (spatial primitives) simultaneously, and to implement their manipulation as transparently as possible (Goodchild, 1987). Thorough research has been carried out on the design of the functionality of GIS software focusing interest on user needs in terms of data volume and manipulation requirements.

Once a system is implemented on this basis, efforts to achieve a predefined classification accuracy may lead to either cumbersome recursion, or overdesigning the capabilities of the system. It seems to be more popular for "GISers" to provide performance tests only in terms of "geometrical representation", however, the community still lacks those tests on matching categorization requirements. Vector viewers specially claim that the raster approach overemphasizes geometric properties, while the vector model permits the attribute to be attached to the appropriate spatial object. Indeed, that is why there is emphasis on the links between geometry and attributes: the appropriate objects are not known *a priori*. It seems to me that the methodology of cartographers has been preferred to modeling uncertainty.

Unless data structures, efficiently handling a set of spatial primitives, are not adjusted to the inherent data characteristics, including accuracy, heterogeneity and the like, there will be no guarantee that a given representation can fulfill the requirements of classification accuracy. On the other hand, whenever the attribute domain was in focus, a very limited set of spatial characteristics, like a single fractal dimension, was taken into consideration (Goodchild and Dubuc, 1987). It would be properly modest to say that we have understood, and more or less successfully modelled, spatial data in the mapping space, while the exploration of feature space is still ahead.

THE CARTOGRAPHIC UNCERTAINTY RELATIONSHIP

The solution of the problem of making reliable maps (i.e. where both locational and classification accuracy is known and limited) has to be accompanied by the recognition that "accurate" and "erroneous" are not just two disjoint sets, but rather should be viewed as a continuum. When map users consider accuracy issues, they certainly want "the best". In simple words, if 10 m and 90% were printed on a tourist map as accuracy limits, they would like to assume that any dark green patch represents a forest with the same locational and thematic accuracy. And this is the point where real data may cause so much trouble to professional modelers. All of our tools dealing with spatial data, and *de facto* our geographical information systems, are context-dependent.

It requires manageable definitions of "objects to be mapped". We may want to ask, for instance in the previous example, whether our definition of a forest is useful at all: Can one, two, three... trees be a forest? Or, if one knows for sure that there is no forest covering more than 10,000 square miles in an area, is that a useful piece of information? Such questions should not look absurd. When soil scientists are calculating the risk of missing a(n infinitely narrow) boundary in the field, landscape architects assign a value of ecological potential for a 5 km * 5 km area, or economists rank countries based on per capita income, etc., they are dealing with very similar problems: Complex human concepts (variables, categories and relationships) are "projected" into Euclidean space in a manner that their potential for further inference is maximized. In other words, spatial homogeneity criteria are defined so that uncertainty is tolerable.

There is a significant mathematical-statistical arsenal to study such criteria. Beyond classical works in autocorrelation studies (e.g. Griffith 1988) more recently attribute classification with spatial constraints has been introduced (Gordon, 1987) more or less independently from mainstream GIS-related research (Chrisman, 1986).

Most importantly to our topic it has been shown for environmental variables that homogeneity criteria based on a given categorization reveal spatial variation (Csillag and Kertesz 1990). Generally speaking, ***there is a contradiction between the requirements of constant attribute accuracy and constant spatial resolution.*** The general concept that fixed these parameters independently over an entire data set cannot be held. If given that recognition probabilities for a class-set are predefined, there is no guarantee that a certain spatial resolution will match any homogeneity criterion. Conclusively, there may not be a unique, generally best resolution for a data set; either accuracy or resolution will exhibit variation.

ECOLOGICAL SITE CHARACTERIZATION - AN EXAMPLE

Let us illustrate the above outlined ideas with a practical environmental mapping example. The task of information processing in this case is to quantitatively describe ecological site characteristics of a salt-affected low-grass prairie (Toth et al., 1990a).

This landscape covers more than 100 km^2 in the Hortobágy-region in E-Hungary, and it can be characterized by abrupt changes in soil conditions, surface grass cover, microrelief with very sharp boundaries (Rajkai et al., 1988). Additionally, the descriptive measures of the apparent surface pattern are highly ***scale dependent***, consequently there have been numerous efforts to determine the spatial behavior of underlying variables. The primary tool of these investigations was geostatistics, but several botanical and cartographic considerations were also taken into account.

The section below is focused on the following problem. Given a set of interrelated variables their spatial characteristics are determined in order to find the most suitable resolution to sample and map them. If these characteristics turn out to be different, a pointwise classification based on these variables will lead to heterogeneous patches. Having a control categorical variable, the spatial variability of the individual variables can be described by patches. How can those patches be found, for which all spatial variances will be lower than an acceptable threshold?

This way one can identify class-membership for any given location with predefined accuracy.

The variables included in this study cover a wide range related to salinity status, soil chemistry, soil texture, etc., as well as microrelief and a number of botanical variables. The typical alkali soils in the Hortobágy National Park, mainly heavy-clay solonetz soils, can be characterized by varying depth of A horizon (Rajkai et al., 1988), and that variation corresponds to the dramatically different surface conditions. It is an erosion process on an almost completely flat plain induced by local disturbance (Toth et al., 1990a). Eroded surfaces occur as micro-valleys, and there is a well-known toposequence from elevated spots through the slopes down to the valleys. As the A horizon is washed away pH, salt-content (S%), and exchangeable sodium percentage (ESP) increase, while ecological diversity decreases. This spatial pattern which is seemingly dominant in the meter range horizontally and in the centimeter range vertically produces a highly complex terrain over the whole extended area.

The mapping strategy must be based on understanding the interrelationships between soils and vegetation forming a complex ecological system. A 15 m by 15 m plot was selected as a test-site for detailed analysis, a number of 60 - 500 m long transects were sampled, while remotely sensed data were collected for regional extrapolation, inventory and monitoring (Toth et al., 1990b). From an environmental point of view the task is to assign description of spatial variation to patches, in terms of variables and resolution, which otherwise would appear as equally homogeneous in terms of salinity status.

Figure 4. summarizes some of the data collected for the test-site. Systematic sampling was applied along the 1.5 m by 1.5 m grid, while stratified random sampling was carried out for the distinct floors of the toposequence, i.e. for hills, slopes and valleys. Soil samples were collected for 100 cm^3 samples, i.e. with approximately 5 cm^2 ground resolution, while botanical data for individual species and total coverage were recorded corresponding to 50 cm by 50 cm quadrats.

The geostatistical evaluation of measurements revealed that there are sharp differences between the spatial characteristics of individual variables, even though they play more or less similar roles in describing salinity status. For example, while pH clearly showed well-defined spatial structure on the test-quadrangle with a characteristic range of about 11 m, that of clay percentage came out to be about 14 m with very high nugget, but salt-content had an unbounded semi-variogram. If one wanted to characterize a given surface within the region, there were always variables, which showed too high estimation variance, or others must have been oversampled. Therefore, an optimum sampling scheme for classification of salinity status based on these variables could not be computed.

Stratified sampling was controlled by botanical data Elevated spots are characterized by more complex associations and more surface cover, while valleys are dominated by one species. This is due to the dramatic difference between their salinity status: Where the A horizon is present, pH and salt-content is lower, while on eroded spots severe salinization

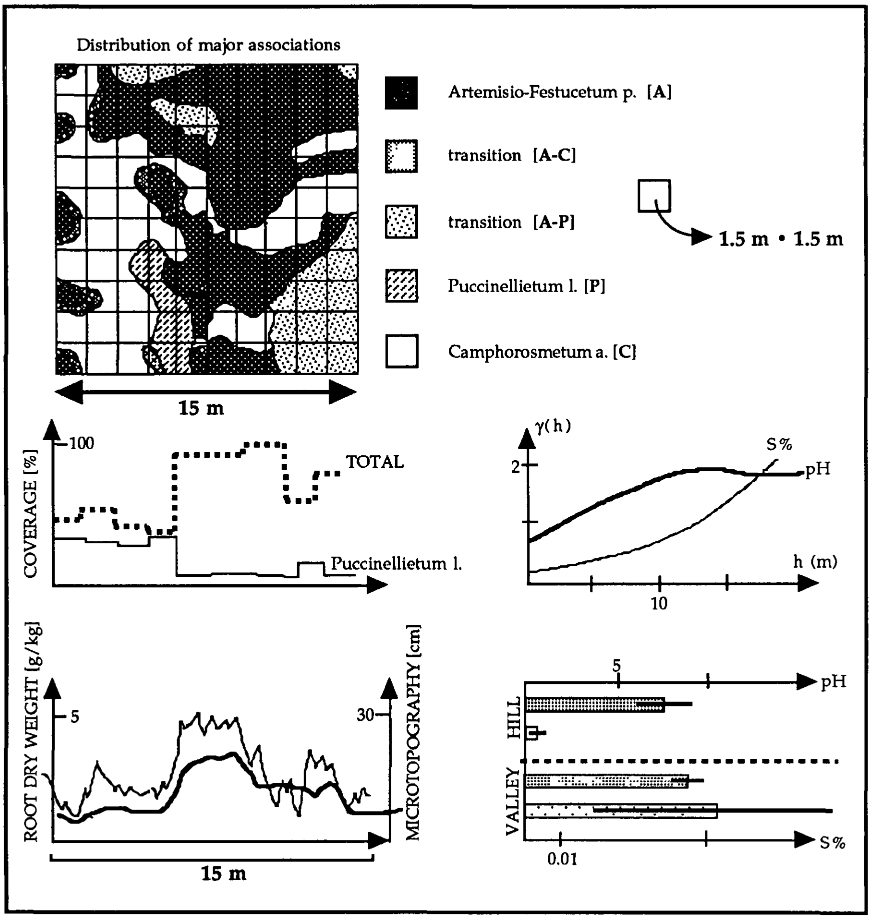


Figure 4.
Environmental data for resolution study (Hortobágy National Park, E-Hungary) - systematic and stratified random sampling
[Distribution and classification of major botanical associations on the test-site (sketch-map, top); Descriptive data along cross transect for vegetation quadrats (top left graph) and for related variables (low left graph); Semi-variograms of two soil properties with curves to guide the eye (top right graph); Descriptive statistics (mean and standard deviation) for pH and salt-content for two-classes of the toposequence (low right graph) - see text for details]

occurs. Although this relationship supported the initial classification, descriptive statistics showed an interesting side-effect. On the hills pH and root-dry weight had significantly higher variance than in the valleys, or on the slopes, while this relationship was reversed in case of, for instance, salt-content. This observation leads again to a conflict, if one wishes to determine the necessary number and distribution of samples to classify a given location.

As a summary of this example the hierarchical nature of the possible solution should be pointed out. On a general soil map this area would be shown as a "highly variable salt-affected" area. Neither does this description contain explicit information about the amount or nature of this variation, nor does it provide reasonable estimates of the key variables by means of descriptive statistics. Having a detailed survey data set, say in a GIS, overlaying pH on salt-content leads to different results depending on which salinity ***class gets preference*** in determining classification criteria. It is because the objects to be mapped, in this case salinity classes, have class-dependent links between the mapping space and the feature space. Therefore, for example, more saline surfaces can be better identified with finer resolution, taking into account more non-spatial variation in salt-content, than non-eroded surfaces, and so on. Furthermore, this information can eventually be incorporated in the data structure as well.

CONCLUDING REMARKS

The evolution of geographical information analysis has resulted in conflicts with the common sense of "resolution". It has been shown that there is inherent uncertainty involved in data models applied in geographical information systems. Several approaches have been applied to spatial data to deal with this uncertainty, but they handle the mapping and feature space separately. In environmental mapping, when resolution of spatial sampling is theoretically unrestricted and classification does not define the spatial objects themselves, the problem of determining ***an*** optimal resolution, which provides a given constant attribute accuracy leads to a contradiction. A soil mapping example outlines that the most promising path for further research is context-dependent merging of criteria defined in mapping and feature space, rather than separating them as independent properties of objects to be mapped. The various statistical tools one can apply through data models permit not only control of accuracy, but they can contribute to the evolution of data structures, which incorporate this information. These data structures should be object-oriented, since there are no objects unless they can be recognized with certain probability, and can be located with certain accuracy.

ACKNOWLEDGEMENT

The research reported here was carried out at the Research Institute for Soil Science and Agricultural Chemistry (MTA TAKI), Hungarian Academy of Sciences before the author joined Syracuse University. The financial support of the National Science Research Foundation (OTKA, Hungary) is gratefully acknowledged. I am indebted for the invaluable discussions with Miklos Kertesz, Agnes Kummert, Laszlo Pasztor, Tibor Toth. Thanks are also due to the two reviewers for comments on the manuscript.

REFERENCES

Bendat, J.S. and A.G. Piersol (1986) ***Random Data: Analysis and Measurement Procedures***; John Wiley & Sons, New York.

Bracewell, R. (1965) ***The Fourier transform and its applications***; McGraw Hill, New York.

Burgess, T.M., Webster, R. (1980) Optimal interpolation and isarithmic mapping of soil properties II. Block kriging; *J.Soil Sci.**31***, 333-341.

Buttenfield, B. (1989) Scale-dependence and self-similarity in cartographic lines; *Cartographica, **26**, 79-100.*

Campbell, J. (1991) ***Map use and analysis***; W.C.Brown Publishers, Dubuque.

Chrisman, N.R. (1986) Obtaining information on quality of digital data; in: *Proc. **AutoCarto London***; Vol.I. 350-358.,

Chrisman, N.R. (1989) A taxonomy of error applied to categorical maps; *Int'l. Cartographic Assoc. World Congress,* Budapest, (manuscript).

Chrisman,N.R. (1989) Modeling error in overlaid categorical maps; in: ***Accuracy of spatial databases*** *(ed. M.Goodchild and S.Gopal),* pp. 21-34.,Taylor & Francis, London.

Clearbout, J.F. (1976) ***Fundamentals of geophysical data processing***; McGraw Hill, New York.

Csillag, F. (1987) A cartographer's approach to quantitative mapping of spatial variability; in: *Proc. **AutoCarto 8**,* pp.155-164.,ASPRS-ACSM, Falls Church.

Csillag, F. (1989) Maps and images preserving the spatial structure of agroecological information; *Proc. RSS Annual Conference (Bristol),* 469-471.

Csillag, F., M.Kertesz (1990) Spatial variability: Error in natural resource maps?; *Agrokemia & Talajtan, **37**,* 715-726.

Dueker, K.J. (1979) Land resource information systems: spatial and attribute resolution issues; *Proc. **AutoCarto IV**,* Vol.II. pp 328-337., ASP-ACSM, Falls Church.

Goodchild, M.F. (1987) Towards an enumeration and classification of GIS functions; in: *Proc. International GIS Symposium (Crystal City),* Vol.II., pp.67-79.

Goodchild, M.F. and O.Dubuc (1987) A model of error for choropleth maps, with applications to geographic information systems; in: *Proc. **AutoCarto 8**,* pp.165-174.,ASPRS-ACSM, Falls Church.

Goodchild, M.F. and S.Gopal eds. (1989) ***Accuracy of spatial databases,*** Taylor & Francis, London.

Gordon, A.D. (1987) Classification and assignment in soil science; *Soil Use and Management, **3**,* 3-18.

Griffith, D.A. (1988) ***Advanced Spatial Statistics,*** Kluwer Academic Publishers, Dordrecht.

Journel, A.G. and Ch.J.Huijbregts (1978) ***Mining geostatistics***; Academic Press, London.

Journel, A.G. (1986) Geostatistics: Models and tools for Earth sciences; *Math. Geol. **18**,* 119-139.

Mardia, K.V. (1980) Some statistical inference problems with kriging II. Theory; in: ***Advances in Automatic Processing and Mathematical Models in Geology***; pp.113-131., SCIENCES DE LA TERRE, Paris.

McBratney, A.B. and R.Webster (1986) Choosing functions for semi-variograms of soil properties and fitting them to sampling estimates; *J.Soil.Sci.*, ***37***, 617-639.

Mark,D.M., F.Csillag (1989) The nature of boundaries in 'area-class' maps; *Cartographica, **26**, 65-79.*

Philip, G.M., Watson, D.F. (1986) Geostatistics and spatial data analysis; *Math.Geol., **18***, 505-509.

Rajkai, K., E.Molnar and J.J.Oertli (1988) The variability of soil properties of a cross-section and its coherence with plant pattern; in: *Proc. XIII. ISSS World Congress Papers (Hamburg),* Vol.III. 1247-1258.

Ripley, B.D. (1981) ***Spatial statistics***; JohnWiley & Sons, New York.

Robinson, A.H., Sale, R.D., Morrison, J.L., Muehrcke, P.C. (1984) ***Elements of cartography** 5th edition* John Wiley & Sons, New York.

Sinton, D. (1978) The inherent structure of information as a constraint to analysis: mapped thematic data as a case study; in:***Harvard Papers on Geographic Information Systems** (ed. G.Dutton), Vol.7.*, Addison-Wesley, Reading.

Tobler, W. (1969) Geographical filters and their inverses; *Geographical Analysis, **1**, 234-253.*

Tobler, W. (1988) Resolution, resampling, and all that; in: ***Building databases for global science** (ed. H.Mounsey and R.Tomlinson)* , pp. 129-137.,Taylor & Francis, London.

Tobler, W. (1989) Frame independent spatial analysis; in: ***Accuracy of spatial databases** (ed. M.Goodchild and S.Gopal),* pp. 115-122.,Taylor & Francis, London.

Toth, T., M.Kertesz, F.Csillag, and L.Pasztor (1990a) From pattern elements toward vegetation processes of continental salt-affected rangelands; *Journal of Rangeland Management* (forthcoming).

Toth, T., M.Kertesz, F.Csillag, and L.Pasztor (1990b) Characterization of semi-vegetated salt-affected landscapes by means of field remote sensing; *Remote Sensing of Environment* (forthcoming).

Vasiliev, I, Frendschuh, S., Mark, D.M., Theisen, G.D. and McAvoy, J. (1990) What is a map?; *Cartographic Journal* (forthcoming)

Webster, R. (1977) ***Quantitative and numerical methods in soil classification and survey;*** Oxford University Press, Oxford.

Woodcock, C.E., Strahler, A.H. (1984) Image variance and spatial structure in remotely sensed scenes; in: *Proc. 2nd NASA Conference on Mathematical Pattern Recognition and Image Analysis,* NASA Johnson Space Center, Houston, pp. 427-465.

Generalization Operations and Supporting Structures

Kate Beard
Department of Surveying Engineering &
National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469
BITNET: Beard@Mecanl

William Mackaness
National Center for Geographic Information and Analysis
Department of Geography
State University of New York - Buffalo

ABSTRACT

Current GIS do not support wide flexibility for the performance of map generalization operations so users have limited opportunity for creating views of data at different levels of resolution. This paper describes a context for computer assisted generalization and reports on a set of generalization operators. The generalization operators are embedded within a larger scheme for a map design system which could be attached to a GIS. The selection and sequencing of operations is not fully automated but relies on user interaction. This approach is adopted to allow users maximum flexibility in tailoring maps to their individual needs. The system, however, is designed to provide substantial support for the user in negotiating this process. The final section of the paper describes data structures for supporting the operations within the context of this interactive environment.

INTRODUCTION

In many studies or projects, we wish to see some piece of geography represented or displayed in a simpler or more abstract form. We may also at any time wish to change the level of detail or level of abstraction of a representation. Although the ability to change the resolution of spatial or non-spatial information in a representation is highly desirable, this capability is not well supported by current GIS. Most commercial GIS software packages support generalization as one or two algorithms for line simplification (João 1990). These systems can be tricked into performing other generalization functions (Daly 1990), but the capabilities are not explicitly documented such that they are readily available to the casual user. The need for flexible and efficient changes in resolution warrants an expansion of generalization capabilities which are easy and intuitive for users to employ. Mackaness and Beard (1990) describe a user interface concept for a map design and generalization system. This paper expands on this earlier concept but focuses more specifically on generalization operations to be included in the system, the context in which they are applied, and proposed structures needed to support them. The paper begins with an overview of the system to provide a context for the generalization operations.

CONTEXT FOR THE GENERALIZATION OPERATORS

McMaster and Shea (1988) and Shea and McMaster (1988) consider the important questions of why, when, and how to generalize. Much of the motivation and selection of type and degree of generalization is driven by user needs and purpose. The remainder is dictated by graphic media and format. This section develops a context for when and how to generalize within the proposed system based on two controlling factors: the user and graphic constraints.

The proposed map design system

The system as proposed by Mackaness and Beard (1990) assumes a vector GIS database exists. Characteristics of this database are described in greater detail in Section 4. It further assumes that users will interact with the database to select and extract information to composé views of the data at different levels of resolution or detail. Generalization operations in this case do not create new databases at coarser resolutions, but create materialized views of the original database. Views have been described in the database literature as an interface between a user (or application) and the database which provides the user with a specific way of looking at the data in the database (Langerak 1990).

In this system, we embed generalization operations within the basic functions of map composition and design. As itemized by Keates (1988) these include

- selection of geographic area,
- selection of information content,
- specification of format,
- specification of scale and
- specification of symbols.

These functions are intricately linked, but not necessarily in sequential order. Although at the outset one would most logically begin with selection of a geographic area, specification of the remaining functions could occur in any order including the ability to revise the size and configuration of the geographic area.

Full automation or system specification of these variables is probably not practical. Map design and generalization decisions depend largely on knowledge of map purpose so user interaction is highly desirable if not required. As Turk (1990) points out, improvements in human computer interaction will require shared cognitive responsibility between operator and computer. The proposed system therefore supports a high degree of user interaction, but is designed to assist the user in navigating through the process. The balance between user specification and system support is based on a consideration of which functions are best handled by the

system, which by the user, and which in some supportive arrangement between the two.
Figure 1 provides an overview of the system with an indication of which steps are user controlled and which are shared or managed by the system. Figures 2a-d illustrate user interface design for specification of the functions shown in Figure 1.

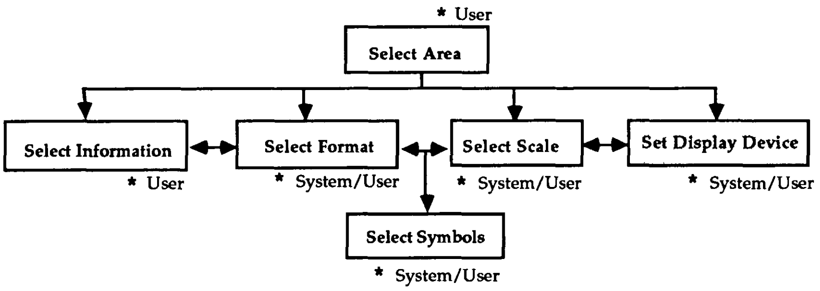


Figure 1. Overview of the system showing relationships between map design functions. Asterisks indicate functions which are controlled by the user and/or the system. There is an implied order to the functions given by the tree structure but the arrows indicate an ability to move freely between the various functions.

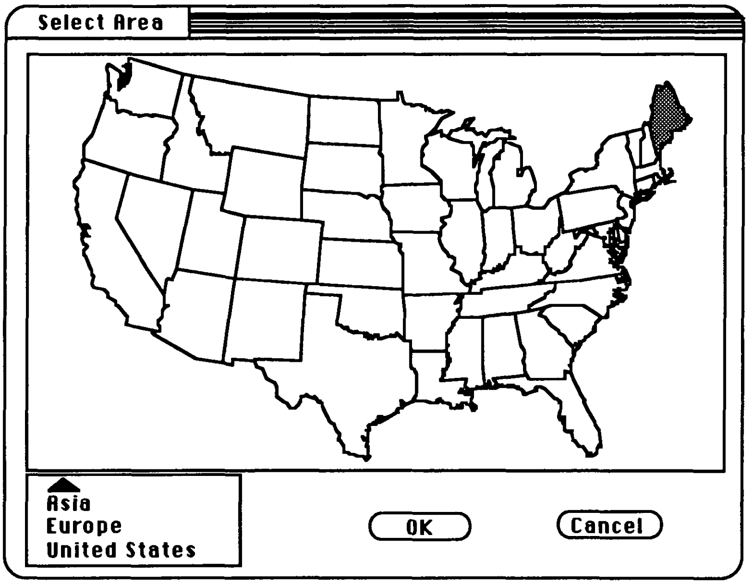
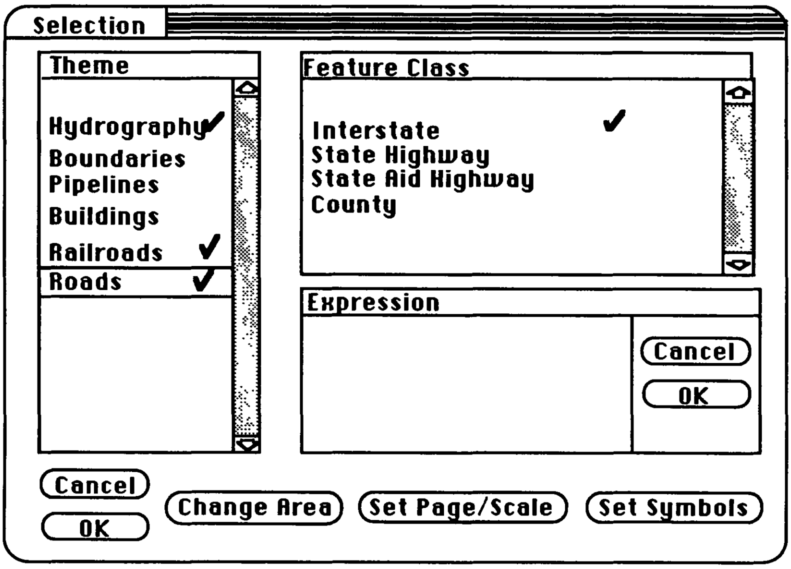


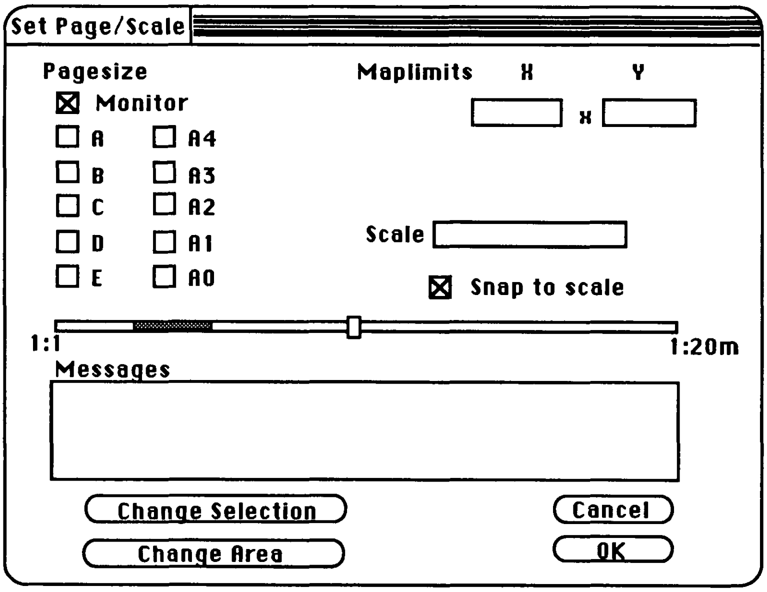
Figure 2 a. Illustration of user interface for selecting geographic area.



Hydrography ✓
Boundaries
Pipelines
Buildings
Railroads ✓
Roads ✓

Interstate ✓
State Highway
State Aid Highway
County

Figure 2b. Illustration of the user interface for selecting information content for inclusion on a map. Buttons on the bottom allow users to move to the other functions/menus.



☒ Monitor
☐ A ☐ A4
☐ B ☐ A3
☐ C ☐ A2
☐ D ☐ A1
☐ E ☐ A0

☒ Snap to scale

Figure 2c. Illustration of the user interface for selecting scale and/or page format. Users can move to the area selection menu or the information content selection menu from this screen.

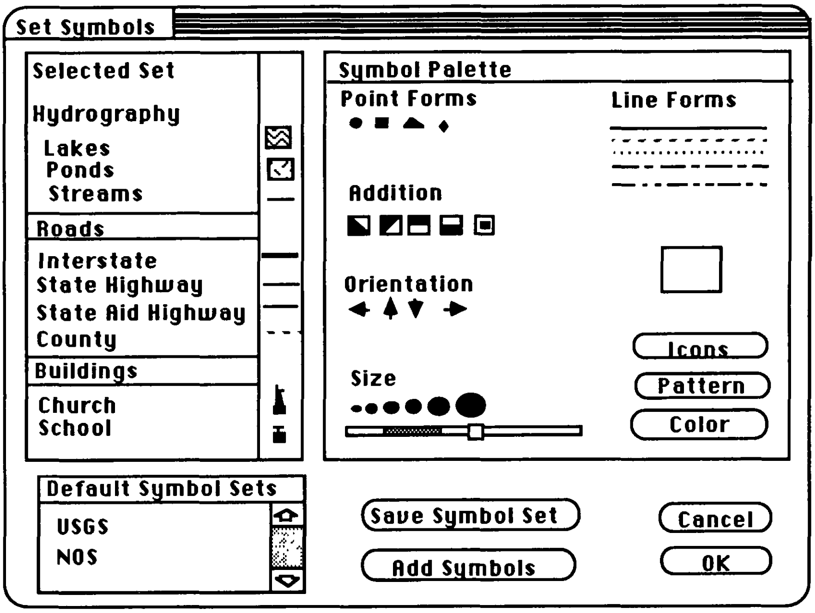


Figure 2d. Illustration of the user interface for specifying symbology. The selected information content is displayed on the left, and a symbol palette for making symbol choices appears on the left. The system indicates an appropriate range of dimensions for symbols given a scale. Selection of a default symbols set is also possible.

Once the user has made preliminary selections, the system can build on this information to provide clues and recommendations for subsequent steps. For example, if a user selects a geographic area which is 4 by 5 miles and selects, as a format, E size paper with a map area of 20 by 24 inches, the system computes a scale. As illustrated in Figure 2c, a computed scale would appear in the scale box and an appropriate scale range would be indicated by the shaded area on the slider bar for selecting scale. Alternatively, if the user specifies an area, information content, and scale, the system can recommend a range of appropriate formats. User specification and system feed back iterate toward an eventual result which meets users requirements and assures a legible display.

Specification of a geographic area, information content, format, scale, and symbology sets the scene for generalization. The combined specification of these five items can generate spatial conflicts or graphic interference, and to create useful and legible products these conflicts must be avoided or resolved. Conflicts can be avoided by re-specifying any one or more of the functions just described or resolved by generalization. In this paper we focus on resolution of conflicts by generalization operations.

Context for identifying and resolving conflicts

The types of conflicts which occur in map design are related to minimum requirements for maintaining graphic clarity and legibility. These minimum requirements have been generally well documented in cartographic texts and cartographic production specifications. They are based on avoiding:

- areas which are too small
- line segments which are too short
- items which are too narrow
- items which are too close.

Items being too close results in congestion, coalescence, or conflict. The result of items being too small is imperceptibility and the same applies to segments which are too short and items which are too narrow. Congestion, coalescence, conflict, and imperceptibility are conditions described by Shea and McMaster (1989) that require some type of generalization for resolution.

These minima can be fixed as thresholds in any appropriate display units (eg. inches as shown in Table 1). Given a specified scale, format, and symbology, items selected from the database for display are screened against these thresholds to identify and locate conflicts. These conflicts are the minimum set of items or features which must be generalized. If any of the specifications are revised, the set of features which must be generalized will change

Conflict	**Threshold**
Too small	.01 sq. in.
Too short	.08 in.
Too narrow	.15 in.
Too close	.20 in.

Table 1. Illustrates fixed thresholds for legibility. In map construction these are transformed according to the selected scale and compared against dimensions of objects in the database.

Assume now the system has identified a list of all features and locations which are: too small, too short, too narrow, and/or too close (includes areas of overlap and coincidence). The specific function of the generalization operators is to resolve these identified problem areas. A set of rules could be formulated to direct the selection and application of generalization operators, but as generalization is intricately tied to map purpose, appropriate operations are difficult to anticipate for all cases. A simple rule would be to omit all areas which are too small. The user, however, may not wish to omit all small features, but exaggerate some or merge them with other nearby objects.

If the desired result is to be achieved from the users perspective, the user must have some involvement in orchestrating the operations. This prompts another balancing of tasks between operator and computer. In this case users are allowed to freely apply operators as they chose, but the system directs them to areas requiring generalization. An important function of the system is to clearly display all conflicts to the user and indicate when they have been resolved. This is handled by two methods. One is by listing objects which are in conflict with themselves or one or more other objects. The other is through graphic display of the conflicts. In the graphic display, all items in conflict (those falling below the thresholds) are displayed in red. All features which can be legibly displayed appear black. The items in red are the conflicts which must be resolved. As conflicts are resolved by generalization operations they are re-displayed in black and their resolution is also indicated on the corresponding tabular listing. Figure 3 provides an example of the interface for displaying conflicts to the user.

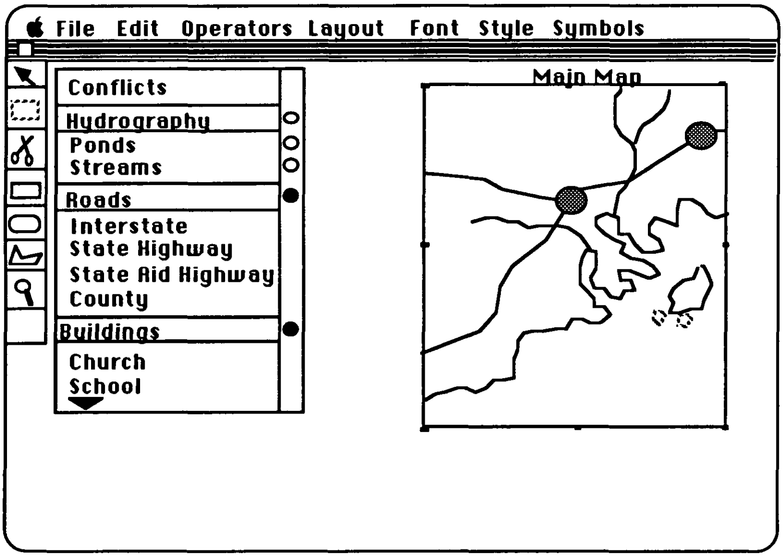


Figure 3. Example of the interface for displaying conflicts to the users. Conflicts are graphically (spatially) identified in red (dashed here) on the map as well as in the listing on the left.

GENERALIZATION OPERATORS

The function of the generalization operators in this context is to adjust a selected set of objects such that they can be legibily displayed at a specified scale, format and symbology. This section identifies a set of proposed generalization operators. Several cartographers have generated comprehensive lists of generalization processes (Steward 1974, Brassel 1985, McMaster and Monmonier 1989, McMaster 1990). Using these inventories, we can apply a structure to assist in identifying an appropriate set of operators. This structure distinguishes between operations on graphic symbols and operations needed to simplify digital representations. These are referred to as structural operators: those that simplify or abstract the level of detail, and display operators: those that adjust the graphic display to ensure legibility. Structural operators can be seen to perform three basic operations: reduction in the number of objects, simplification of spatial detail, and simplification of attribute detail, with combinations of the three possible (Beard 1990). Display operators include operations such as displacement, masking, and symbol changes needed to resolve symbol collisions when a representation is displayed. This structure can be applied to McMaster's (1990) list of operations, for example, to assemble a toolbox of operators. For this system, operations from each category are selected to provide users a range of options and tailored for the purpose of resolving the conflicts identified above.

Proposed generalization operators
In this system we include the following operators:

Operations which reduce the number of objects
- select
- omit

Spatial operators
- coarsen
- collapse
- combine

Attribute operators
- classify

Display operators
- exaggerate
- displace

The names of many of these operators have appeared in the literature previously (Shea and McMaster 1989, Nickerson and Freeman 1986, Brassel 1985, Lichtner 1979), but their functions may differ here to specifically respond to conflict resolution. The functions of these operators as used in this system are described below.

SELECT: This is a special operator which must precede all others. It is required to initialize the composition of a graphic view of the database which can then be displayed on a monitor or as hardcopy output. The user is informed of information stored in the database and from this they may select items by theme, feature type, or instance (see Figure 2b). This operation allows the user to explicitly choose only desired items. For example, the user may select the theme roads, in which case all roads in the selected geographic area will be extracted for display. The user may also be more specific and select only Interstate Highways or to be most specific, select only Interstate 95 for example.

OMIT: Once items have been selected for display, the omit operator allows removal of objects. These objects are only removed from the display list and not from the database. As with the SELECT operator, individual objects may be removed or objects may be removed by theme, feature type, or conflict type. For example the OMIT operator could be used to remove all objects which were too small.

COARSEN: This operator removes fine spatial detail (crenellations from a line). This operator could be applied to objects stored in the database with a high level of spatial detail, and which the user wishes to display in less detail. This operator works primarily on metric detail, but may change the topology of objects. Figure 4 illustrates an example of application of this operator to a lake with an island. In the resulting figure, the metric detail has been modified and the island has been removed, changing the topology.

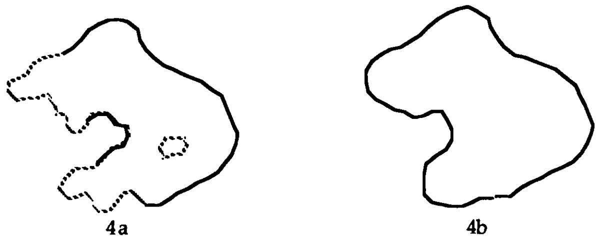


Figure 4a. shows a lake with an island at the level of detail it is stored in the database. Figure 4b show the same lake after application of COARSEN. The areas in conflict are show by dotted line (in color, these would be shown in red). In the resulting figure the conflicts have been resolved.

The user need not specify parameters for this operator. They only need select the object or objects to be coarsened and apply the operator. The operator uses the minimum thresholds which have been computed for the selected scale or format. The resulting representation is therefore appropriate to the selected scale. As shown in Figure 4, the small bays and island which fall below the threshold for areas too small, items too close, or items too narrow are removed by the coarsen operator. This operator can be applied to individual objects, themes or feature types.

COLLAPSE: The collapse operator substitutes a 1D or 0D representation for a 2D representation. This operator could be applied to objects stored in the database as areas, but which a user wishes to display as points or lines. Figure 5a and 5b show examples of COLLAPSE as applied to an estuary and a city. This operator must be preceded or succeeded by a symbol change. COLLAPSE resolves the legibility problem of items being too close, or COLLAPSE followed by a change in symbol width could resolve the problem of items being too small or too narrow.

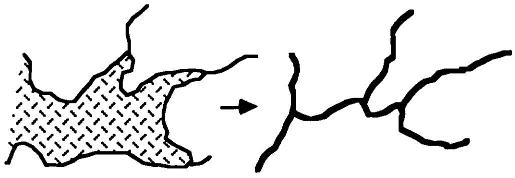
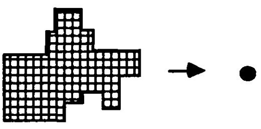



Figure 5a. COLLAPSE applied to an estuary
city.

Figure 5b. COLLAPSE applied to a

COMBINE: The combine operator simplifies a spatial representation by merging objects which are nearby in space into a single new object. For example a cluster of small islands may be combined to form a larger island. The operator applies only to two or more selected objects and the result is always one new object. Thus COMBINE is strictly a localized operator. This operation must be preceded or succeeded by the CLASSIFY operator so that the resulting object is properly identified. Figure 6a and 6b illustrate an example of COMBINE. COMBINE resolves items being too small or too close.

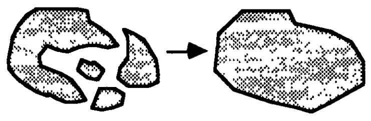


Figure 6a. COMBINE applied to islands.

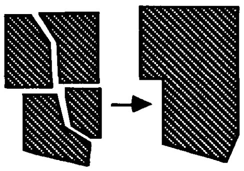


Figure 6b. COMBINE applied to fields.

AGGREGATE: This operator is similar to COMBINE but merges objects which are adjacent rather than those with intervening spaces. CLASSIFY must precede this operator as well. The aggregate operator can be applied globally by theme or by feature class.

CLASSIFY: This operator allows individual objects, feature types or themes to be assigned to a new class. The classification may be based on shared attribute characteristics of objects. The user or systems selects a set of objects and assigns a new class label (eg. For all objects with attribute D, Class = M). A symbol change must follow this operation, and when the new symbol is assigned, all objects assigned to the new class inherit the symbol. This operator does not directly resolve conflicts but is required as a supporting operation for operators which change the nature of an object (i.e. COMBINE and AGGREGATE).

EXAGGERATE: The exaggerate operator expands the size or width of objects. It can be applied by theme, feature type, instance or conflict type. The operator expands the object to meet the minimum threshold for legibility and therefore requires no parameter specification by the user. For a line or point representation, the width or radius is expanded. This can be accomplished by redimensioning a symbol. For an area, the operation performs a localized scale increase.

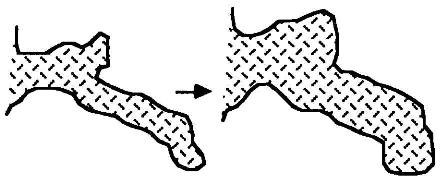


Figure 7a. EXAGGERATE applied to an inlet

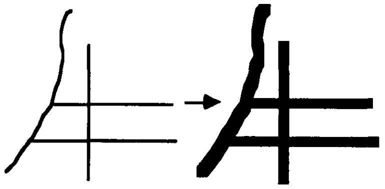


Figure 7b. EXAGGERATE applied to roads

DISPLACE: This operator is applied locally to two or more objects which are too close or overlapping.

Conflicts can be resolved by several different generalization operators with choice dependent on the desired outcome. Objects which are too small can be resolved by omitting them, exaggerating them, or combining them with other nearby objects. Objects which are too close can be resolved by omission, collapse, simplification, combination, or displacement. The selection and application of the operators is left to the user to allow them the most freedom in constructing a map to fit their needs. Some order is imposed in that some operators will not be accessible depending on the state. For example, SELECT is the only operator which can be accessed initially, and AGGREGATION may not be applied without first applying CLASSIFICATION.

Another key aspect in the design of operators is that they obey one overall rule. That is they are to resolve one or more conflicts when invoked and create no new conflicts. This rule is used to avoid convoluted iterations of operations in the resolution of conflicts. In particular this implies that all symbol specification occurs prior to generalization. For clearly, if symbol re-dimensioning occurs subsequent to generalization operations, new conflicts will arise and the generalization must be renegotiated.

SUPPORITNG STRUCTURES FOR GENERALIZATION OPERATIONS

For effective interactive use of the system, two tasks in particular must be performed efficiently. Conflict areas need to be identified rapidly so users can be quickly informed of the number and location of conflicts. Secondly the operators themselves must perform efficiently. In this section we consider supporting structures for facilitating each of these tasks.

Conflict Identification

Section 2 identified four types of conflicts. The first was areas too small to be legible. Identification of these conflicts is relatively straightforward. We first assume that areas are computed and stored as attributes of closed polygonal objects. Then, once a scale has been specified or computed, the minimum area threshold is derived, and conflicts are returned from the boolean function:

```
AREATOOSMALL = OBJECTAREA ≤ THRESHOLD

If AREATOOSMALL then DISPLAY (OBJECT, RED)
```

The number of comparisons required is of order N, the number of polygon objects selected for display.

Identification of the remaining conflicts depends on finding the euclidean distances within and between objects that are smaller than the minimum threshold computed from scale and symbol dimensions. This three additional boolean functions:

SEGTOO SHORT = SEGLENGTH ≤ SEGTHRESHOLD

WIDTHTOONARROW = OBJECTWIDTH ≤ WIDTH THRESHOLD

TOOCLOSE = POINTTOPOINT ≤ CLOSETHRESHOLD

To support these functions, we could conceivably pre-compute and store all distances between objects (objects in this case being points) as an ordered list. Discovery and retrieval of all violating objects could then follow by a search using distance as the key through the set of records ordered by distance between and within objects. This approach is sufficient to identify conflicts and provide the information to display conflict areas. The cost of computing and storing distances, however, is too high to justify simply the identification of conflicts. On the other hand if the cost can be spread over several other operations it becomes more justifiable. Our second criteria was to support efficient performance of generalization operations. In the next section we examine how pre-computed and stored distances figure into the resolution of conflicts and performance of the generalization operators.

Data Structures and Operator Performance

The number of operations dependent on knowledge of distance between objects implies the need for a database organized by spatial proximity. Such databases have been previously researched (Matsuyama 1984, Samet 1984) and arguments made for their use in the context of map design and generalization (Mackaness and Fisher 1987). Matsuyama's method, however, does not explicitly represent distance relationships among objects. Vornoi diagrams and the dual Delauney triangulation have also been proposed for representing spatial proximity relationships (Green and Sibson 1977, Brassel 1978, Gold 1987, 1989), but these also do not implicitly or explicitly store a full complement of distance relations. In Figure 8, form triangle edges we could derive distances from P6 to P4-P9 but not directly to P1 or P11.

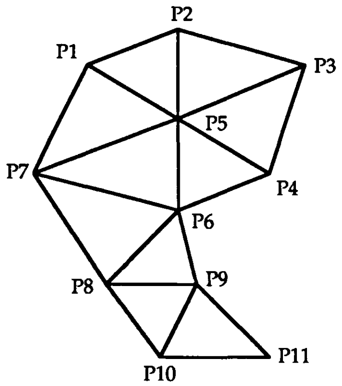


Figure 8. Distance relationships in Delauney Traingles.

In most cases, queries to these structures can return a spatial neighborhood or the set of objects within a neighborhood. Distances can then be computed for these smaller sets. In this system, access to distance relationships is required frequently and uniformly over a geographic area. Given the level of interaction, the system also requires fast performance. The COARSEN, DISPLACE and EXAGGERATE operators in particular can benefit from immediate access to stored distance relationships. The next section describes a structure for storing and retrieving distance relationships. It assumes distances between points have been pre-computed.

A data structure for storing and retrieving distances.
Recall that the function of the generalization operators is to resolve identified conflicts and create no new conflicts. To assure that conflicts are resolved and no new ones created requires knowledge of distances between and within objects. Operators therefore need information beyond an ordered list of distances sufficient for identifying conflicts. In this case we need to know not just that a distance is sub-threshold but the locations where sub-threshold distances occur. The search condition thus involves the combination of three keys (Distance, X and Y), creating a multidimensional or range query problem. Assuming a threshold distance T, what we are after is a piece of the XY plane that yield clusters of points less that T distance apart.

The structure required is an indexed sequential data structure. Such a data structure accommodates both random and sequential access to records. In this case we adapt a method described by Orenstein and Merrett (1984). This involves interleaving bits of the tuple (DIST, X, Y) and storing the 'shuffled' tuples in the database. Interleaving the bits of a tuple maps a k-d space (3 in this case) to a 1-d space, creating a Z-ordering. The Z-ordering assures that points which are close in k-d space will be close in 1-d space. A similar ordering was first used by Morton (1966) for CGIS and has been replicated and expanded since by several others (Bentley 1975, Burkhardt 1983, Orenstein 1983, Ouksel and Scheuermann 1983, Tropf and Herzog 1981).

As Orenstein and Merrett (1984) note, the domains of the attributes in the tuple need not be the same size. An array [attr] can be used to indicate the attribute from which each bit was taken, yielding the shuffle function h(t) = [attr]{i} = i mod k where t is any tuple and k is the number of attributes per tuple.

Each bit in the 'shuffled' tuple corresponds to a split of a region of the three D space into two subregions of equal size. The bit equals 0 for one subregion and 1 for the other. Each additional bit splits the previous two subregions into two sub-sub-regions and so on. The direction of the split is given by the attribute [attr] from which the bit originated.

Sub-regions can be described by prefixes of the shuffled value. The addition of bits to the prefix refines subregions as described above. Smaller prefixes in other words correspond to larger pieces of the XY plane and larger distances.

Information retrieved from this structure can support identification of conflicts and provide direct input for the COARSEN, DISPLACE and EXAGGERATE operators. We discuss retrieval next in the context of the COARSEN operator.

Information is retrieved from the structure by a 3d search region SR. Initially SR is the entire space. A query region QR is posed given by the minimum bounding rectangle (MBR) of an object selected for COARSENing and by the threshold TOOCLOSE. If SR is outside QR, then SR contains no tuples satisfying the query and no action is required. If SR is inside QR, all points in SR satisfy the query and are unshuffled and returned. If SR overlaps QR but is not within it, SR is split into two new SRs. This step is applied recursively until SR is within QR. Several SRs may be required to cover a given QR , a weakness of this scheme which Orenstein and Merrett note. Once the final set of SRs is determined, tuples are actually retrieved by both random and sequential access. To use Orenstein and Merrett's notation $SR_{lo} : SR_{hi}$ denotes a range of shuffled values corresponding to a prefix. Retrieval of all points from an SR requires retrieving the tuples (t) such that $SR_{lo} \leq \text{shuffle(t)} \leq Sr_{hi}$. The data structure can be randomly accessed using SR_{lo} as the search argument. Then sequential accesses retrieve tuples until the shuffle value of a tuple exceeds SR_{hi}.

The set of tuples (DIST, X, Y) returned by this search procedure provide direct input for COARSEN. COARSEN performs a cluster analysis on the returned points and distances. The outcome of the cluster analysis is a reduction in the number of points such that no two are closer than threshold T. Simplified objects are then recomposed from the remaining points (see Figure 4).

A similar retrieval of records supports DISPLACE. DISPLACE is a localized operator applying to a small area. The area in which displacement will occur can be selected by clicking and dragging to define a rectangle. This rectangle and threshold TOOCLOSE define the query region QR. The search procedure returns the set of points within the rectangle and the distances between them.

SUMMARY

This paper discusses the context for a flexible and interactive approach to generalization. The design of the system seeks a balance between user responsibility and discretion and system intelligence to assist the user. The user makes initial selections for geographic area and information content. They may also specify scale, format and symbology or allow the system to compute or set defaults. Four types of graphic conflicts are identified as arising from these specifications. The selected objects can be too small, too short, too narrow or too close for the given scale and symbols dimensions. The purpose of generalization operators is to resolve such conflicts and assure a legible display. Identification and location of conflicts requires knowledge of distances between and within objects.

Distance computations are costly no matter how they are approached, but they are critical to operation of the system. The high degree of interaction demands high performance from the system. To support efficient interaction, we investigated methods for pre-computing and storing distances. An indexed sequential data structure is proposed to support efficient retrieval of information, but this must be subjected to testing to assure adequate performance.

ACKNOWLEDGEMENTS

This work has been supported in part by National Science Foundation grant SES-88-10917 and represents a contribution to Research Initiative 3, "Multiple Representations" of the National Center for Geographic Information and Analysis. The NSF support is gratefully acknowledged.

REFERENCES

Beard, M.K. 1990. 'Constraint Based Transformation for Map Generalization' NCGIA Symposium, "Towards a rule based symposium for map generalisation" Syracuse, NY, To appear in Map Generalisation: Making Decisions for Knowledge Representation London: Longmans (forthcoming).

Beard, M. K. 1988. Multiple representations from a detailed database: a scheme for automated generalization PhD thesis, University of Wisconsin, Madison.

Beard, M.K. 1987. How to Survive on a Single Detailed Database. Proceedings Auto Carto 8. pp. 211-220.

Bentley, J.L. 1975. 'Multidimensional binary search trees used for associative searching' Communications of ACM 18:9 pp. 9-517.

Brassel, K. and Weibel, R. 1988. 'A Review and Conceptual Framework of Automated Map Generalization,' International Journal of Geographical Information Systems. 2: pp. 229-244.

Brassel, K.E. 1985. 'Strategies and Data Models for Computer-Aided Generalization'. International Yearbook of Cartography. 25: pp. 11-30.

Brassel, K. 1978. 'A topological data structure for multi-element map processing', In Proceedings of the First International Advanced Symposium on Topological Data Structures for Geographic Information Systems (G. Dutton, Ed) VOL. 4. Addison-Wesley, Reading, MA.

Burkhardt, W.A. 1983. Interpolation Based Index Maintenance.' BIT 23:3 pp. 274-294.

Daly, R. 1990. 'Map Generalization using ARC/INFO' Research Report No. 10. Northwest Regional Research Laboratory, Lancaster University.

Gold. C. M. and Cormack, S. 1987. Spatially ordered Networks and Topographic reconstruction. International Journal of Geographic Information Systems. 1: pp. 137-148.

Gold, C.M. 1989. 'Spatial Adjacency: a General Approach. Auto Carto 9. pp. 298-312.

Green, P. J. and Sibson, R. 1977. 'Computing Dirichlet Tessellations in the Plane', Computer Journal. 21:2 pp. 168-173.

João, E. M. 1990. 'What experts systems don't know: the role of the user in GIS generalization'. Proceeding NATO ASI, on Cognitive and Linguistic Aspects of Geographic Space. Las Navas del Marques, Spain.

Keates, J. S. 1989. Cartographic Design and Production, New York, NY. Longman Scientific and Technical.

Keates, J. S. 1982. Understanding Maps London: Longman.

Mackaness, W. and Beard, K. 1990. Development of an Interface for user interaction in rule Base Map Generalization. Technical Papers GIS/LIS '90. Anaheim, CA. 1: pp. 107-116.

Mackaness, W. A. 1990. 'Application and evaluation of generalization techniques' NCGIA Symposium, "Towards a rule based symposium for map generalisation" Syracuse, NY, To appear in Map Generalisation: Making Decisions for Knowledge Representation London: Longmans (forthcoming).

Mackaness, W. A. 1988. Knowledge based resolution of spatial conflicts in digital map design Unpublished PhD Thesis, Kingston Polytechnic, UK. May 1988.

Mackaness, W. A. and Fisher, P. F. 1987 'Automatic recognition and resolution of spatial conflicts in cartographic symbolisation' Auto Carto 8 Baltimore, Maryland. pp. 709-718.

Matsuyama T, Hao, L.V. and Nagao, M., 1984. 'A file organisation for geographic information systems based on spatial proximity', Computer Vision, Graphic and Image Processing. 26:3. pp. 303-318.

McMaster, R. 1989. 'Introduction to Numerical Generalization in Cartography', In Numerical Generalization in Cartography. Monograph 40. Cartographica. 26: 1. pp. 1-6.

McMaster R. B. and Shea, K.S. 1988. 'Cartographic Generalization in a Digital Environment.: a Framework for Implementation in a Geographic Information System'. Proceedings GIS/LIS '88. San Antonio. 1: pp. 240-249.

Morton, G.M. 1966. 'A computer oriented geodetic database and a new technique in file sequencing.' Unpublished manuscript, IBM, Ltd. Ottawa Canada.

Muller J C 1989. 'Theoretical considerations for automated map generalisation', ITC Journal 3:4. pp. 200-204.

Nickerson, B. G. and Freeman, H. 1986. 'Development of a Rule-Based System for Automated Map Generalization,' Proceedings, 2nd International Symposium on Spatial Data Handling. pp. 537-556.

Orenstein, J.A. 1983. 'A Dynamic Hash File for Random and Sequential Accessing'. In Proceedings of the 6th International Conference on Very Large Databases. Florence, Italy. IEEE, New York, pp. 132-141.

Orenstein, J. A. and Merrett, T.H. 1984. 'A class of data structures for associative searching'. In Proceedings of the 3rd ACM SIGACT- SIGMOD Symposium on Principles of Database Systems. Waterloo, Ontario. ACM, New York, pp. 181-190.

Ouksel, M. and Scheuermann, P. 1983. 'Storage mappings for multidimensional linear hashing' In Proceedings of the 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. Atlanta, GA. ACM, New York, pp. 90-105.

Peckham, J. and Maryanski, F. 1988. 'Semantic Data Models,' ACM Computing Surveys. 20:3. pp. 153-189

Robinson, A. H. , Sale, R, Morrison, J. L. and Muercke, P. 1984. Elements of Cartography 5th Edition New York, NY. John Wiley and Sons.

Shea, K. S. and McMaster, R. 1989. 'Cartographic Generalization in a Digital Environment: When and How to Generalize', Auto Carto 9. pp. 56-65.

Steward H J 1974 'Cartographic generalisation: some concepts and explanations' Cartographic Monograph No 10 Toronto: University of Toronto Press.

Topfer , P. and W. Pillewizer . 1966, 'The Principles of Selection', The Cartographic Journal, 3:1. pp. 10-16.

Tropf, H. and Herzog, H. 1981. 'Multidimensional range search in dynamically balanced trees'. Angew Info. 2: pp. 71-77.

Turk, A. 1990. 'Towards an understanding of human computer interaction aspects of geographic information systems' Cartography (in press).

USING SPLINE FUNCTIONS TO REPRESENT DISTRIBUTED ATTRIBUTES

John R. Herring
Intergraph Corporation
Huntsville, Alabama 35894-0001 USA
e-mail: ingr!b17a!mobius!jrh@uunet.uu.net

ABSTRACT

All current GIS systems assign discrete, static attribute values to geometric objects (vector, pixel, or voxel). This is not how the world usually works. Physical objects of geographic importance are heterogeneous things. The width, depth, and flow-rate of a river, the porosity, density, and permeability of a rock body, the pressure, temperature, and velocity of the air or water, all of these things vary in complicated, sometimes chaotic, and convoluted ways; ways that affect our experience and ways that would effect our computer models, if we took them into account, and knew how to deal with them. Given this fact-of-life, the next generation of GIS systems must have a mechanism to model truly continuously variable attribute values. Spline functions gives us one such a way.

Spline functions have long been used in CAD/CAM to represent geometric forms, curves and surfaces, a use that they are well qualified to perform in GIS applications (see for example Auerbach (1990)). But splines are a much more general concept than a convenient way to store geometry; they are a way to efficiently approximate, to any degree of accuracy, any function. By shifting our paradigm, we can make the dimensions of the splines simultaneously represent both geometry and attribute distributions.

INTRODUCTION AND BACKGROUND

Any information system must be able to model the reality of its application (Casti (1989)). A database designer begins with a methodology (for example, entity, attribute, relation modeling), that at an abstract level, uses a model of reality onto which he will impose his data concepts by a series of data transformations, eventually mapping the highest level abstractions by stages to a concrete storage mechanism (*Date (1983)*, Ullman (1988) and (1989), Codd (1990)). This resultant storage mechanism unfortunately puts restrictions back upon the scope of the original abstract model, often restricting the attributes of a data item to the fundamental data types of integer, floating point, character string and variants thereof. In addition, the data types are usually considered independent of the methods needed to manipulate them, leaving the application the requirements to supply not only ingenious storage work arounds, but also the edit, analysis, query and mechanisms needed. This compounds the fundamental database management system problems of data integrity and semantic data control (see Özsu (1991)). With the advent of abstract data types (ADT), this is no longer the case (Gorlen (1990)). Using ADT's, the database designer can encapsulate a complex data storage format with the methods for its creation, manipulation, analysis, query, and display. This process is beginning to make its way into commercially available relational data bases (RDB) such as Empress, Oracle, Informix, and Ingres, and is the foundation of the new object oriented technology, such as Versant (VERSANT Object Technology), and Ontos (Ontologic) (see Khoshafian (1990)). This paper investigates the use of spline functions as an ADT for the storage of both space varying and time varying attributes.

Spline functions (see Farin (1990), Bartels (1987), Faux (1979)), often used in computer aided design and manufacture (CAD/CAM), are actually part of a very old branch of mathematics, approximation theory. Basically, splines allow us to approximate any function by the specification of a set of control points in the range of the function (called "poles", not necessarily function values) which control a varying weighted average based upon a set of functions (called "weight functions"). In CAD/CAM applications, the poles are 3-D points, and the weight functions map a compact subset of a Euclidean space (of dimension 1, 2, or 3) to the unit interval $[0,1] = \{ x \in R \mid 0 \leq x \leq 1\}$. The resulting range of the spline is a geometric object (contained in the convex hull of the poles). This object is either a curve, surface, or solid depending upon the dimension of the domain space; 1, 2 or 3 respectively.

More simply put, this paper proposes that we take the CAD/CAM interpretation of a spline, and extend the dimensions of the both the domain (source) and range (target) space; so that a CAD/CAM 3D point (x, y, z) becomes a GIS N-D point (x, y, z, time, density, porosity, permeability, ...). This approach basically generalizes the use of geometry to represent geography into the use of geometry to represent any measurable quantity; an old, well known and understood concept that most people encounter in a first algebra course.

The remainder of this paper develops the theory of splines to support this concept and gives examples of the type applications most likely to useful in GIS applications. Anyone wishing to learn about splines for their own sake is directed to the references, especially Farin (1990) which presents a more geometric development than most, and Auerbach (1990) which is a good example of the use of splines in geographic visualization. The development presented in this paper will emphasize some particular aspects of splines in ways peculiar to their use in supporting the spatial and temporal distribution of attributes.

FUNCTIONAL REPRESENTATIONS

A functional geometric description used in CAD/CAM is a generalization of the algorithmic construction objects used in vector data sets (line, polyline, polygon, circles, ellipse, general conics, etc.). Each functional geometric object consists of a domain (source or parameter space) and a function mapping the domain into the range coordinates system, usually E^3, 3-dimensional Euclidean space. Domains are usually some subset of a Euclidean space, the most common of which are I (the closed interval from 0 to 1, [0,1]) or some unit cube I^n (the cartesian product of I with itself, n times).

Spline functions are defined as a variable weighted average, using weight functions, over the domain of some specified set of points in the target coordinate space (poles) (see Farin (1990), Rogers (1990), Bartels (1987) or Faux (1979)). Formally, for a three-dimensional data set, we have a (interval) spline as a function:

$$f : I^n \rightarrow E^3$$
$$f(t) = \sum_{i=1}^{n} w_i(t)\, P_i$$

where t is a vector in I^n, P_i are points in E^3 and $w_i()$ are functions from I^n into I^1 such that:

$$0 \leq w_i(t) \leq 1 \text{ for every } t = (t_1, \ldots, t_n) \in I^n$$
$$\sum_{i=1}^{n} w_i(t) = 1$$

The affect of any one of the poles P_i is felt only where the associated weight function w_i is non-zero (called the support of w_i).

Geometrically speaking, the weight functions are usually bell-shaped curves with a single maximum point (the parameter value of which is usually a "knot" associated to the pole), tapering off to 0 in all directions away from this central peak. Because of this, the poles of a spline are often near critical points of the spline, often the value of the spline as evaluated at the knot. A spline passes through a particular pole, in general, only if the associated weight function is 1 at its knot value (which implies that all other weights are zero).

On I^n, the most common splines are based upon the n'th tensor product of weight functions for I. Given collections of weights, w_i and W_j we can define a collection of $w \otimes W$ by $(w \otimes W)_{i,j}\,(u,v) = w_i(u)\, W_j(v)$. This set of functions can be used as weights since I is closed under multiplication and

$$\sum_{i,j} (w \otimes W)_{i,j}\,(u,v) = \sum_i \sum_j w_i(u)\, W_j(v) = \sum_i w_i(u) \sum_j W_j(v) = 1 \times 1 = 1$$

Splines built using such tensor product weight functions are tensor splines. Most commercially available packages use exclusively tensor splines for higher dimensional functions due to their ease of computation, see Faux (1979).

Generalizations of these standard cubes can involve the choice of a different interval to support either computational convenience or added geometric interpretations of the parameter; for example it is often computational advantageous to use time or arc length for curves (discussed below), see Farin (1990). Unless otherwise stated, we will assume that the parameter cubes I, I^2, ... , I^n can be based upon any intervals in Euclidean space, as needed to support interpretations.

Some earlier nontensor higher dimensional spline work used triangles in place of cubes producing what is called a simplical spline (see Farin (1990)). This type of splines has generated some interest in the GIS applications, specifically in contour preserving surface visualization using a simplical decomposition or triangulated irregular network (TIN) based upon a constrained Delaunay triangulation, in Auerbach (1990).

A special case of the spline function is the B-spline. B-splines use piecewise polynomial or rational functions for weight functions. Each weight function's support spans an interval defined by a set of knots (the number of which is the order of the spline). This gives the spline designer a "local control" that allows him to adjust pole values while only affecting the spline is a very restricted neighborhood of the pole's knot. Further, given a set of sample points (t_i, v_i) $t_i \in I$, $v_i \in E^n$, $1 \le i \le m$, there are closed form solutions to finding m poles for which the associated spline exactly fit the samples, or for finding least-square "best fit" splines with a fewer number of poles (see Bartels (1987)). All of this discussion can be generalize to TIN's and to general simplical complexes (see Farin 1990).

Derivatives

It should be noted here that, while not always precisely spline functions themselves, the various derivative of a spline have easily calculable forms. Given a differential form D (i. e. something like $\frac{\partial}{\partial t_k}$, $1 \le k \le n$), the value of the form applied to a spline can be expressed as (a result of simple calculus):

$$D\, f(t) = \sum_{i=1}^{n} [\, D\, w_i (t)\,]\, P_i$$

In all cases, this is not a spline function as it is written (the sum of the derivative weights would necessarily be a constant 0, since $\Sigma\, D\, w = D\, (\Sigma\, w) = D\, (1) = 0$), but calculations of the various D f() is not significantly harder than the calculation of the spline values themselves. Further, for particular classes of splines, such as Bezier splines, there is collection of poles that will represent D f() as a spline function of a different degree (see Faux (1979)).

Curves

Curves can be represented as one-dimensional splines:

$$c : I^1 \rightarrow E^3$$

The continuity and differentiability of the curve are determined by the smoothness of the weight functions. The various derivatives of the curve as a function have exactly what you might expect. c'(t) is a tangent vector to the curve, with magnitude equal to the velocity of "t" with respect to arc length (thinking of t as a time component). c''(t) is acceleration, with a component parallel to the curve (parallel to c'(t)) giving the acceleration of t with respect to arc length, and the remaining component vector normal to the curve pointing directly away from the center of curvature. The component of c'''(t) perpendicular to the c'(t), c''(t) plane is a binormal indicating the direction of the torsion (twisting) of the curve (tendency of the curve to leave a planar surface) (see Rogers (1990)).

Using different parameterizations, gives some other interesting physical interpretations to c'() and c"(). If c() is parameterized by the arc-length (usually written as "s") of the resultant curve, then the c'(s) is the unit tangent and c"(s) is in the normal with the length of c"(s), written as $\|c(s)\|$, being the local Gaussian curvature of the curve c(t), the inverse of the radius of curvature. The acceleration vector, c"(t), will always line in the plane of c'(s) and c"(s), so that we could write

$$c''(t) = a\, c'(s) + b\, c''(s)$$

where "a" is the magnitude of the force of acceleration along the curve, and "b" is the magnitude of the force of acceleration due to change of direction (a sort of steering force).

Surfaces

Surfaces can be represented as two-dimensional tensor splines:

$$s : I^2 \rightarrow E^3$$

The images of the domain lines in the surface give a spline grid of constant parameter values. The partial derivatives of a surface spline give us the tangents to the surfaces in the direction of the associated parameter curves. Another interpretation of a surface tensor spline is as a parameterized set of curve splines. Assuming that we have a surface spline, s(t, u), we can define

$$\forall u \in I, \quad c_u(t) = s(t,u).$$

Applying our knowledge of curves, we know that

$$c_u'(t) = \frac{\partial}{\partial t} s(t, u)$$

And, swapping the roles of u and t, we also have

$$c_t'(u) = \frac{\partial}{\partial u} s(t, u)$$

Interpreting this in terms of the geometry, we can say that the partial derivatives of the surface spline are tangent vectors to curves totally contained within the surface. Assuming that the surface spline function is well behaved, the two tangent vectors give us a spanning set for the plane tangent to the surface at the corresponding point.

Simplical splines are closely related to triangulated irregular networks (TIN). Based upon a triangulated domain, the most common methodology would be to use the underlying geographic surface as the spline's domain. The use of a generalized full 3D triangulation would allow the surface to fold back over itself by allowing multiple s values for a single (x,y).

Volumes

Solids (volumes) can be represented in two distinct manners. The most common representation is as a collection of surfaces which form the boundary of the solid. In terms of distributions of attributes, this formulation would be useless, since it does not distribute the parameter space into the interior of the volume. In the second technique, generalizing from the above, we can always consider solids as representable as three-dimensional splines:

$$v : I^3 \rightarrow E^3$$

Such volume splines must usually be almost everywhere one-to-one to prevent the function from collapsing multiple points from the parameter space into single points in the range space (the mathematical equivalent of "spindle, fold and mutilate"). The most common exception to this is where the boundary of the parameter space is collapsed to give us non-rectilinear ranges.

If we apply the same technique to the parameters of a volume as we did above to the surface, we can view the function as a parameterized set of surfaces or as a 2-parameter set of curves. The embedded surfaces are called a "foliation" of the volume.

The generalization of the TIN based spline uses simplices of dimension 3 (tetrahedrons), see Herring (1990). For geographic use, the underlying tetrahedron irregular network would normally be a simplical complex spanning the volume of interest.

Higher Order Geometries

All of the above geometric descriptions can be extended to a 4th or higher dimension entity using the same techniques. An interesting hybrid is to use temporal spline curves $P(t)$ to describe the motion of the poles of a spline through time. For the tensor splines, this is simply going to a spline of one higher dimension. For a simplical spline, this forms the tensor product of the existing spline with the temporal curve, as opposed to forming a 4D simplical complex. Even for simplical splines, tensoring with time curves is probably the preferable technique, since this most closely matches the way in which one thinks of time and motion. For most applications where time can be treated as an independent dimension, this technique should be applicable without much difficulty.

DISTRIBUTIONS

Distributions of attribution can be addressed by spline and other functional representations in two basic manners. The first technique includes the definition of the attributes with the geometry in a single spline function, The second technique uses multiple splines over a single parameter space. Other approaches can be viewed as combinations, and multiples of these first two.

In the first approach, given an attribute or a set of "k" attributes, each of which is expressible as a real value, and each of which is a continuous function of space, we can generate spline functions whose first three range coordinates (dimension of target geometry) represent points, and whose k trailing range coordinate values are for the attributes along the spline:

$$f : I^n \rightarrow E^3 \text{(geometry)} \times E^k \text{(attributes)}$$

This generalizes to temporal variability through the use of a space time geometric component, giving us:

$$f : I^n \rightarrow E^3 \text{(geometry)} \times E^1 \text{(time)} \times E^k \text{(attributes)}$$

In the second approach, the attributes are generated by separate spline functions, but sharing a common parameter domain. Thus, we have a set of functions, f_0, f_1, etc.. The first functions gives us a mapping from the parameter space to the geometry, and each additional function generates a single attribute or a set of related attributes. The value of an attribute "a" at a point is then given by an implicit equation:

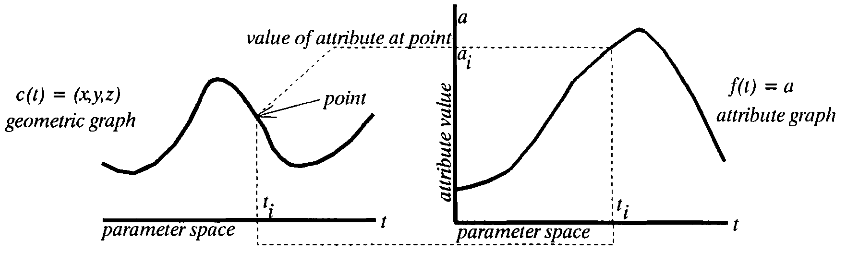
$$\forall\, t \in I^n \quad f(t) = \text{value of attribute a at the point } f_0(t)$$

In using splines or other functional geometric descriptions for the distribution of attribute values, we are creating a tensor sum model that makes no implementation distinction between geometry and other numerically measurable attributes (Herring (1990)).

Distributing Attributes Along a Line Feature

To distribute an attribute along a line feature, two pieces of information are needed. First, we need a parameterization of the line to use to associate spine values to positions on the line. Second, we need a set of sample values of the attribute along the line, or a mechanism to generate those values. Putting these pieces of information together, we now have a set of sample pairs consisting of the parameter values and attribute values:

$$S = \{(t_i, a_i) \quad t_i \in I,\ a_i \text{ the attribute value at the point on the line associated to } t_i\}$$



The General Approach

We now have to choose a set of poles P_j and weights $w_j : I \rightarrow [0,1]$ ($1 \leq j \leq m$), that will generate a spline function

$$f = \sum_{j=1}^{m} w_j(t) P_j$$

such that:

$$\forall (t_i, a_i) \in S, \ f(t_i) \cong a$$

To associate a spline function to an existing line, we have to define how the parameter space is mapped to positions along the curve.

The spline case: If the line is already a spline (geometric) we can use this geometric spline's parameterization.

If our samples are at knots in the spline's parameter space, then we can augment the existing geometric poles with and additional dimension for the attribute, adjusting the pole-attribute values (a'_j), until the sample attribute values are achieved. This gives a combined geometric-attribute representation, as follows:

$$\forall t \in I, \quad f_0(t) = \sum_{i=1}^{m} w_i(t) \, (P_i, a'_i)$$

Assuming that we have n such geometrically correlated attributes, we have an extended spline function as follows:

$$\forall t \in I, \quad f_0(t) = \sum_{i=1}^{m} w_i(t) \, (P_i, a'_{1,i}, a'_{2,i}, \dots, a'_{n,i})$$

Where P_i is the original geometric pole, and each $a'_{k,i}$ is an appropriately chosen value so that the k'th attribute value is achieved at the i'th knot.

If the attribute values are statistically independent of the shape of the geometry, or we do not have attribute values for the knots of the geometry spline, then the above method will not work. But using the same parameterization, we can define separate splines for each attribute or set of correlated attributes, using only the common parameter space to synchronize curve geometry and attribute distribution. Given a set of sample values (x_i, y_i, z_i, a_i), $1 \leq i \leq k$, of the attributes along the line feature, using the geometry spline $f_0()$, we solve for t_i such that:

$$f_0(f_i) = (x_i, y_i, z_i)$$

This gives us a set of spline functions samples (t_i, a_i), which we can use to generate a spline (using weights "W") that precisely fits the samples with k poles, or an approximation with fewer poles, giving us a spline $f_1()$ such that:

$$\forall t \in I, \ f_0(t) = \sum_{j=1}^{m} w_j(t) \, P_j$$
$$\text{and} \ \ f_1(t) = \sum_{i=1}^{n} W_i(t) \ a'_i$$
and the value of attribute "a" at $f_0(t)$ is $f_1(t)$.

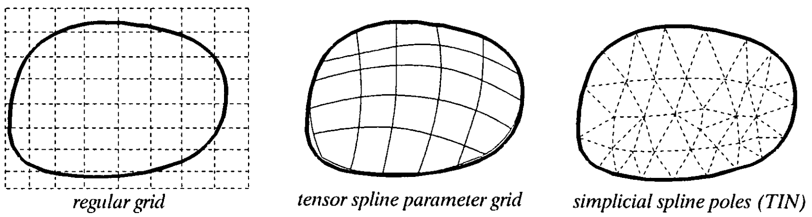
Multiple attributes can be handled either in single splines (as was done with the geometry spline above or as separate splines.

The Polyline and General Case: If the geometry of the line does not come with its own parameterization, then we can use any function, such as arc-length of a point from the line's beginning, as a distribution parameter. Using arc-length as a parameter defines the domain interval of the spline as I = [0, L] where L is the total length of the line feature. Using this parameter, we are essentially in the second case from the section above. The new equation for the t_i is:

$$t_i = \text{distance along the line from start position to } (x_i, y_i, z_i)$$

Distributing Attributes Through an Area Feature, Across a Surface

The area, or surface distribution problem revolves around a restriction on the types of spline functions used for higher dimension. Most common software packages use tensor product splines. This places a restriction on the types of knot spacings that can be used. If k_i is the knot associated with $w_i()$ and k'_j is the knot associated to $w'_j()$, then the knot associated to $W_{i,j}() = w_i \otimes w'_j()$ is (k_i, k'_j). This means that the knots are geometrically dispersed in the parameter space in rows and columns (possibly nonuniformly spaced). There are three basic alternatives: 1) use a regular geometric grid as a spline parameter domain, 2) use a tensor spline geometric description of the area, or 3) use a simplical spline.



Pole Geometries for Alternative Distibution Types

If we chose to use a regular grid parameter space, we would create orthogonal profiles in a coordinate block large enough to encompass the entire area, associating the grid points internal to the area to interpolated attribute values, and external grid points to extrapolated attribute values. Here the basic problem is the "regularization" of the data to the grid points (discussed below).

In either of the other two alternatives, there are two problems. First, we have to disperse knots and geometric poles to describe the surface (x,y,z) or area feature (x,y). Second, we have to obtain attribute values for the points on the surface associated to each of the knot pole pairs. Having these, we can apply the algorithms described above to obtain attribute values for the poles that will give us the required distribution function.

Picking the Grid Points: If we already have a spline representation of the surface and the attribute values for the corresponding points on that spline surface, the simplest solution is to use the geometric knots (a direct analogy to the line cases).

If we do not have a spline surface and we wish to use a tensor spline, we can create a pseudo grid across the feature by digitizing two sets of profile lines, cross-hatching the area, using the intersections of these profiles to associate to a similarly set of orthogonal profiles in the chosen parameter space, creating a tensor spline surface that approximated the area feature. Using this method would not necessarily obtain a spline surface whose edge exactly matched the boundary of the boundary of the delineated area (splines can be made to fit a finite number of points, not usually an entire curve). The accuracy of the fitted surface would be a function of the complexity of the area boundary, and the order and number of poles of the chosen spline, but as long as the new surface covered the area feature, every point in the area would have an associated attribute value by the resultant spline distribution.

Alternately, if simplical splines can be used, a tessellation of the surface can be made using the Delaunay (or other) triangulation of the input attribute data samples.

Regularization of the data: In either or the grid techniques, it is probable that after getting a spline approximation of the area, the attribute values for the points on the spline surface will have to be approximated. Various such approximation techniques exist. Using the Delaunay triangulation of samples and either linear or "stolen area" interpolation (Gold (1989) and (1990)), simplical splines (Auerbach (1990)), kriging (Journel-78, David-76), and cokriging are good examples. The interpolation scheme may be chosen depending upon the particular application or depending on a priori assumptions about the data. Recall that the knots, or weights used for the geometric approximation need not be the ones used for the attribute approximation, as long as the parameter space is the same. In the simplical spline case, assuming the data points were chosen with care, little or no interpolation of Pole values should be necessary.

Distributing Attributes Through a Volume

The volume case is similar to the area case, except that a 3-dimensional approximating spline, a 3-dimensional regularization technique, or a 3-dimensional tetrahedron irregular network, as appropriated, are needed.

Vector Fields, Differential Equations and Trajectories

The use of splines to represent vector fields, and the ability to take derivatives of splines leads to their use to represent differential equations and systems of differential equations. For example, suppose that we have a spline representation of current flow in a hydrologic system. Thus, we have a function $F(u,v,w) \longrightarrow (x,y,z,dx,dy,dz)$ that maps a three dimensional parameter space into position and velocity. We can define a solution, or trajectory, to the differential equation:

$$(c,c') = F$$

as a function $c(t) \rightarrow (x,y,z)$ as one such that:

$$c(t) = \pi_{x,y,z} F(u,v,w) \Rightarrow c'(t) = \pi_{dx,dy,dz} F(u,v,w)$$

where "π" is the projection onto the subscripted coordinates.

A NOTE ON EXPERIMENTATION

Much of what is presented here can be classified as speculative, and in a normal situation, I would have waited for until more experimental results in specific applications could have been simultaneously reported. I choose not to delay for a variety of reasons. First, a great deal of work has gone into the various geometric aspects of spline curves and surfaces and, in a very real and meaningful way, this paper is simple a reinterpretation of those results. For example, Auerbach (1990) could be interpreted to show results on the distribution of a single attribute value over an area feature; its geometric representation (graph) resulting in a surface – contours representing isoclines. Secondly, a large part of this paper is a survey of some simple mathematical truths, viewed from an unusual perspective. Unlike physical science, most mathematical papers do not require experimental results to be valid. Third, and most important, is the potential scope of the applications of this sort of technology is broad enough to require multiple efforts to validate it. For example, the distribution of attributes along lines may solve the dynamic segmentation problem in road maintenance systems. The distribution of attributes in areas has applications in any field which needs to represent heterogeneous dispersions; forest or soil management, ecological applications such a predator prey simulations, etc.. Splines have the potential of solving some of the data volume problems associated to grid based map algebra systems. In 3 dimensions, spline distributions have a great deal of potential in representing heterogeneous aggregate both in geology and in engineered materials.

Given the potential of spline distributions and the track record of splines in the geometric applications, it seemed that the probability for successful experimentation in a wide variety of potential applications is very high.

SUMMARY AND IMPLICATION

Spline functions can be used to approximate a large variety of attribute distributions, through any standard geographic feature, to any accuracy or representation quality required. The implications of the methods outlined here are far reaching.

They can change the way we think of attribution. Attributes need not be thought of as static constants, but can be set to vary of both time and space. Attributes can include complex mathematical structures such as vector fields, set of trajectories for differential equations, etc.. Such attributes can be represented to any degree of accuracy required via the use of standard spline functions.

They can solve some long standing storage problems. Spline functions are known to be extremely efficient storage mechanisms, requiring as little as a tenth or a hundredth of the space as compared to vector representations of equal accuracy and quality of representation and visualization.

B-splines and NURBS (non-uniform rational b-splines), which are a standard in CAD applications and deliverable as standard software packages, meet the accuracy and representation requirements of these geographically and temporally distributed attributes. As a software engineering bonus, common geometric representations such as splines simplify system development.

Simplical splines solve some of the problems found in the standard tensor splines, and are a mechanism to visualize distributions from raw sample data. Theoretically, they should have many of the advantages of TIN based DTM's over grid representations.

ACKNOWLEDGMENT

I would like to thank Andrew Frank, Thomas Fisher, and Eric Bayer for their participation in the free form technical discussions over a period of year which lead directly to this paper. A good idea can have many fathers; a bad one is always an orphan.

REFERENCES

Auerbach, S. and Schaeerben (1990), "Surface Representations Reproducing Digitized Contour Lines", Mathematical Geology, Vol. 22, No. 6, pp 723-742.

Bartels, Richard H., John C. Beatty, Brian A. Barsky (1987), An Introduction to Splines for use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers, Inc., Los Altos, California

Casti, John L. (1989), Alternate Realities: Mathematical Models of Nature and Man, John Wiley & Sons; New York.

Codd, E. F. (1990) The Relational Model for Database Management: Version 2, Addison Wesley Publishing Company, Reading, Massachusetts.

Date, C. J. (1983), An Introduction to Database Systems, Vol. II (printed with corrections 1985), Addison Wesley Publishing Company, Reading Massachusetts.

Date, C. J. (1984), Relational Database: Selected Writings, Addison Wesley, Reading, Massachusetts.

Date, C. J. (1986), An Introduction to Database Systems, Vol. I, Fourth Edition, Addison Wesley Publishing Company, Reading Massachusetts.

David, M. (1976), "The Practice of Kriging", Advanced Geostatistics in the Mining Industry, M. Guarascio, et al. editors, D. Reidel Publishing Company, Dordrecht, Holland

Farin, G. (1990), Curves and Surfaces for Computer Aided Geometric Design, Second Edition, Academic Press, Boston, Massachusetts.

Faux, I. D., and Pratt M. J. (1979), Computational Geometry for Design and Manufacture, Ellis Horwood Limited, Chichester, UK.

Gold, Christopher M. (1989), Chapter 3 - Surface Interpolation, Spatial Adjacency and GIS, in Three Dimensional Applications in Geographic Information Systems, ed. J. Raper, Taylor and Francis Limited, London, pp. 21-35.

Gold, Christopher M. (1990), Spatial Statistics Based on Voroni Polygons, paper presented at the Atlantic Institute Seminar, August 1990, University of New Brunswick, Fredericton, New Brunswick, Canada, Y. C. Lee coordinator.

Gorlen, Keith E., Sanford M. Orlow, and Perry S. Plexico (1990), Data Abstraction and Object-Oriented Programming in C++, John Wiley and Sons, Inc., Chichester, UK.

Herring, John R. (1991), The Mathematical Modeling of Spatial and Non-Spatial Information in Geographic Information Systems, David Mark and Andrew Frank, ed., Cognitive and Linguistic Aspects of Geographic Space, Proceedings of a NATO Advanced Study Institute, (to appear).

Journel, A. and C. Huijbregts (1978), Mining Geostatistics, Academic Press, London, England.

Khoshafian, Setrag and Razmik Abnous (1990), Object Orientation: Concepts, Languages, Databases, User Interfaces, John Wiley and Sons, Inc., New York.

Munkres, James R. (1963), Elementary Differential Topology, Princeton University Press, Princeton, New Jersey.

Özsu, M. Tamer and Patrick Valduriez (1991), Principles of Distributed Database Systems, Prentice Hall, Englewood Cliffs, New Jersey.

Rogers, David F. and J. Alan Adams (1990), Mathematical Elements for Computer Graphics, Second Edition, McGraw Hill Publishing Company, New York.

Stonebreaker, M., ed. (1986), The INGRES Papers: Anatomy of a Relational Database System, Addison Wesley, Reading, Massachusetts.

Ullman, J. D. (1988), Principles of Database and Knowledge-Base Systems, Vol I, Computer Science Press, Rockville, Maryland.

Ullman, J. D. (1989), Principles of Database and Knowledge-Base Systems, Vol II, Computer Science Press, Rockville, Maryland.

NEW PROXIMITY-PRESERVING ORDERINGS FOR SPATIAL DATA

Alan Saalfeld
Bureau of the Census[1]
Washington, DC 20233

ABSTRACT

This paper presents new methods for ordering vertices or edges in a tree (connected acyclic graph). The new orderings are called *tree-orders*; they can be constructed in linear time; and they are fully characterized by a useful proximity-preserving property called *branch-recursion*. The paper describes how *tree-ordering* techniques can be applied to find orderings for other types of spatial entities:

- ordering points in the plane,
- ordering points in higher dimensional spaces,
- ordering vertices of any graph,
- ordering edges of any graph,
- ordering line segments in two-dimensional networks,
- ordering line segments of networks in higher dimensions,
- ordering regions in the plane,
- ordering $(n-1)$-cells in n-dimensional polytopal regions,
- ordering n-cells in n-dimensional cell decompositions.

For each of the spatial entities listed above, the orderings produced by extending the tree-ordering methods exhibit important proximity-preserving properties. The paper includes a description of several potential applications of the new orderings of the diverse spatial objects.

PRELIMINARIES

At its most elementary level, database management is the art of organizing or ordering data so that they may be accessed and utilized most efficiently for some particular set of operations of interest. This paper presents a new way of ordering data that will permit a collection of important operations related to clustering and to systematic sampling to be carried out efficiently and effectively. In this

[1]The views expressed herein are those of the author and do not necessarily represent the views of the Bureau of the Census.

section we review and summarize some definitions and basic concepts needed to describe our ordering techniques.

Orderings and Lists

Throughout this paper, *ordering* or *order*, without any qualifying adjectives, will refer to a total order or linear order of a finite set of n elements. Such an order is nothing more than a sequencing of the n elements, a one-to-one association of the elements of the set with the integers 1 through n, or a listing of the n elements. A set of n elements that have been ordered will be called a *list* or an *ordered list.*

In a list of n elements, the $(i + 1)$st element is the *successor* to the ith element; and every element except the nth or last element has a unique successor. Similarly, every element except the first element has a unique *predecessor.* We may build a *cyclic list* or a *cyclic order* from a list by naming the first element to be the successor to the last element (and the last element to be the predecessor to the first). Cyclic lists are often useful because they have no distinguished elements that require special case handling. For example, with a cyclic list, one may begin *anywhere* in the list and exhaustively enumerate elements by taking successors until one returns to the chosen starting element.

Spatial Queries

Points in two-dimensional and higher dimensional space are often assigned an order or primary key to facilitate their storage in and retrieval from databases. Space-filling curves, such as the Peano key and Hilbert curve ([FAL1] and [FAL2]), have proved quite useful for range queries and nearest neighbor queries. These curves are instances of a large class of orderings called *quadrant-recursive* orderings ([MARK]). The defining property of quadrant-recursive orderings is that, in any recursive decomposition of a rectangular region into subquadrants, the points of any subquadrant always appear consecutively in the quadrant-recursive ordering. Points within any subquadrant are enumerated exhaustively before exiting the subquadrant. We will see in the section on branch-recursion that quadrant-recursive orderings are a special case of a more general class of orderings called *branch-recursive.*

Systematic Sampling

Systematic sampling traditionally refers to selection of a subset from a list, where the subset is formed by selecting elements at regular intervals (called the *skip interval*) [KISH]. Elements may be *weighted* to adjust their probability of selection (see figure 1).

If all weights are 1, then a skip interval of k produces a $1/k$ sample. We may think of the sampled elements as having the induced order of their sequenced systematic selection achieved by skipping through the list.

If points in the plane are assigned any quadrant-recursive order, then a systematic selection procedure will sample every subquadrant, no matter what its size, to within one unit of the overall sampling fraction. This representative coverage property was noted and utilized for Peano key ordering by Wolter and Harter [WOLT].

If we regard systematic sampling as a means of ordering subsets, then trivially we may recover the original order of an unweighted list by sampling with skip interval equal to 1. This seemingly trivial means of recovering an ordering will be exploited in the section on tree-ordering vertices to build an ordering when

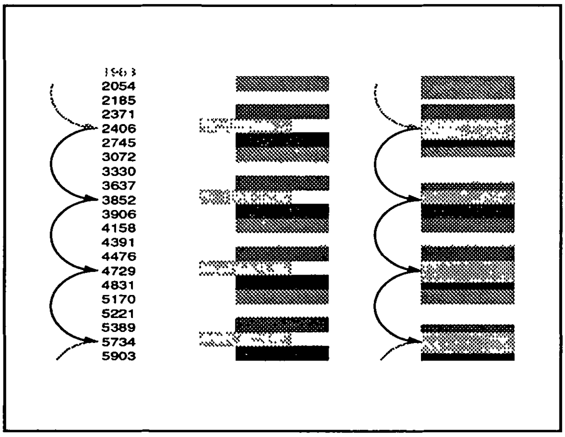


Figure 1: Systematic Sampling from Lists

we sample a set systematically *after* breaking each element into weighted pieces having total weight 1.

Graphs and Maps

The linework of any map has an underlying structure of a *graph*[2]. We will use the usual combinatorial definitions of graph theory found in the standard text by Harary [HARA]. A *graph* $G = (V, E)$ consists of a finite non-empty set V of *vertices* together with a set E of unordered pairs of vertices called *edges*. A vertex v and an edge $\{u, w\}$ are *incident* if and only if $v = u$ or $v = w$. The *degree* of a vertex is the number of edges incident to the vertex. A *walk* of the graph G consists of a sequence $(v_1 v_2 v_3 \cdots v_k)$ of vertices v_i, not necessarily all distinct, such that for each $j = 1, 2, \ldots, (k-1)$, $\{v_j, v_{j+1}\}$ is an edge of G. A *tour* is a walk $(v_1 v_2 v_3 \cdots v_k)$ such that $v_1 = v_k$. A *path* is a walk with no edges repeated. A *cycle* is a path $(v_1 v_2 v_3 \cdots v_k)$ with $k \geq 3$ such that $v_1 = v_k$. A *tree* is a graph with no cycles. A tree as we have defined it is sometimes called a *free tree* to differentiate it from a *rooted tree*, which possesses a distinguished vertex called the *root*.

PROPERTIES OF TREES

We describe some properties of trees that make them easier to work with than graphs in general. We will show in the sections on ordering vertices and edges in a graph how problems of ordering graph components can be converted to problems of ordering tree components for a derived tree. Computer scientists have developed a number of ways of ordering the vertices of rooted trees embedded in the plane [AHO2]. We will be looking at new orderings for free trees.

Combinatorial Properties

We list some important properties of trees.

[2]For some applications it may be preferable and even necessary to regard the linework of a map as a *pseudo-graph*, a structure which allows multiple edges between two vertices. For the applications which we are examining here, the distinction is unimportant.

Property 1 *Every tree with n vertices has exactly $(n-1)$ edges.*

Property 2 *A connected graph having n vertices and $(n-1)$ edges is a tree.*

Property 3 *Adding a new edge to a tree (between existing vertices) always creates a cycle.*

Property 4 *Removing an edge always disconnects a tree.*

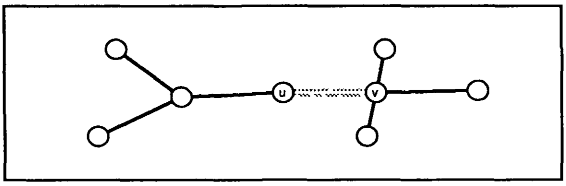


Figure 2: Edge Removal Creates Two Branches

We say that each edge determines two *branches* which are the disconnected component subtrees resulting from that edge's removal. Always one of the branches determined by the edge $\{u, v\}$ contains u and the other branch always contains v.

Planar Embeddings

Not every graph can be drawn in the plane with non-intersecting line-segment edges, but a tree can always be represented or realized as a straight-line drawing in the plane. Moreover, suppose that for each vertex in a tree, we arbitrarily assign a cyclic order to the edges incident to that vertex. Then there is always a drawing in the plane of that tree with straight-line-segment edges such that the clockwise order of the edges incident to any vertex is the arbitrarily assigned cyclic order of the edges about the vertex.

EULERIAN TOURS

An Eulerian tour of a tree is a special well-balanced tour that traverses every edge exactly twice, once in each direction. We give two equivalent descriptions of an Eulerian tour of a tree. ***Each description depends on our having assigned a cyclic order to the incident edges of each vertex.***

Geometric Version

Draw the tree so that the assigned cyclic order of edges at each vertex is the clockwise order. Start a tour at any vertex x. Depart along any incident edge $\{x, u\}$ toward u. Upon arriving at u, depart along the edge $\{u, v\}$ that is next to $\{x, u\}$ in the clockwise order around u. Upon arriving at v, depart along the edge $\{v, z\}$ that is next to $\{u, v\}$ in the clockwise order around v, etc., until finally you return to x along the edge that precedes $\{x, u\}$ in the clockwise order (see figure 3).

The tour that we have described will traverse each edge twice, once in each direction and visit every vertex a number of times equal to its degree.

This tour can also be visualized as follows: imagine that the tree itself is the top view of a wall. Walk next to the wall with your right hand continuously

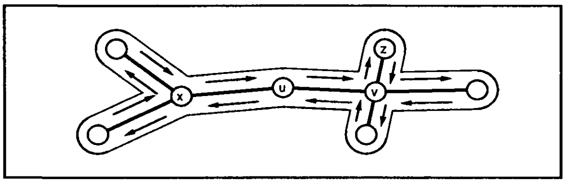


Figure 3: Geometric Depiction of Eulerian Tour

touching the wall. You will eventually return to your starting point, at which time you will have touched both sides of every wall. Thus, had someone else been walking on top of the wall and keeping up with you, that person would have walked every edge of the tree exactly twice.

Combinatorial Version

An Eulerian tour of a tree on n vertices is a walk $(v_1 v_2 v_3 \cdots v_{2n-1})$, where the v_i's are vertices, clearly not all distinct, satisfying

1. $v_1 = v_{2n-1}$.
2. For $i = 1, 2, \ldots, 2n-3$, $\{v_{i+1}, v_{i+2}\}$ is the successor to $\{v_i, v_{i+1}\}$ in the cyclic ordering of edges about the vertex v_{i+1}.
3. $\{v_1, v_2\}$ is the successor to $\{v_{2n-2}, v_1\}$ in the cyclic ordering of edges about the vertex v_1.
4. For every edge $\{u, v\}$ of the tree, the sequence uv and the sequence vu each appear exactly once in the walk $(v_1 v_2 \cdots v_{2n-1})$.
5. Each vertex except v_1 appears as often as its degree.
6. The vertex v_1 appears one more time than its degree.

Notice further that if uv appears before vu in the Eulerian tour, then the subwalk between uv and vu that starts and ends at v completely consumes every edge and vertex in the v-branch determined by edge $\{u, v\}$. Moreover, this subwalk touches nothing but the v-branch of the tree (see figure 4).

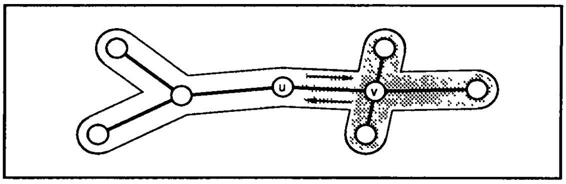


Figure 4: A Subwalk Consumes an Entire Branch

Similarly, the remainder of the tour (the part before uv and after vu) completely consumes every edge and vertex of the u-branch resulting from the removal of the edge $\{u, v\}$. Clearly, no vertex or edge that appears in the v-branch can ever appear in the u-branch.

We summarize this partitioning of the tour into two non-intersecting walks in the following lemmas.

Tour Splitting Lemmas

Lemma 1 *Let $(\cdots x \cdots uv \cdots y \cdots vu \cdots z \cdots)$ be an Eulerian tour of some tree, with x, y, z, u, v vertices on the tour. Then $u \neq y$, $x \neq y$, and $z \neq y$.*

Lemma 2 *Let $(\cdots u_1 v \cdots x u_2 \cdots)$ be an Eulerian tour of some tree, with $u_1 = u_2$ and no other occurrences of u_1 between u_1 and u_2. Then $v = x$.*

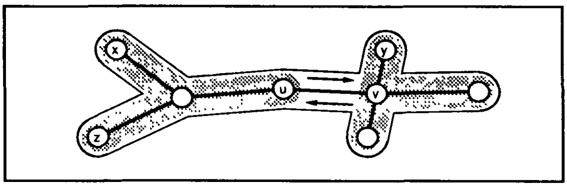


Figure 5: Edge Removal Splits the Tour

Figure 5 illustrates the proof of both lemmas. The subwalks uv and vu are the only means of getting from one branch to the other.

Eulerian Tree Orderings

During the course of an Eulerian tour, all of the vertices and edges are visited at least once. Suppose we wish to assign the integers 1 through n to our n vertices. A procedure that visits the vertices in Eulerian tour order, assigning either the next available number or no number to every *visit* of each vertex, in such a way that exactly one of the visits of each vertex receives an order number, will be called an *Eulerian tree-ordering* (ETO) of vertices.

A procedure that visits the edges in Eulerian tour order, assigning either the next available number or no number to every *visit* of each edge, in such a way that exactly one of the two visits of each edge receives an order number, will be called an *Eulerian tree-ordering* (ETO) of edges.

Garey and Johnson [GARE] and others [PREP], [EDE1] describe one such Eulerian tree-ordering of vertices of a Euclidean minimum spanning tree (EMST) obtained by starting anywhere on the Eulerian tour and assigning the next available number to the *first* visit to each and every vertex (see figure 6).

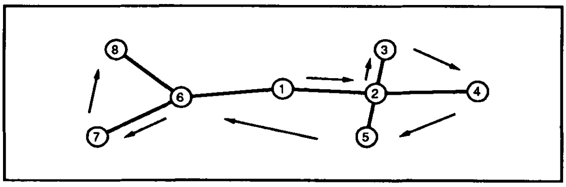


Figure 6: First-Visit Eulerian Tree-Ordering

Their ordering, or for that matter, any other Eulerian tree-ordering of a EMST will always approximate a Euclidean Travelling Salesman Tour to within a factor of 2, since the Eulerian tour itself is never more than twice the length of the Euclidean Travelling Salesman Tour.

We now describe a simple procedure for generating all other Eulerian tree-orderings of the vertices of a tree.

Suppose that we are given a tree *and an Eulerian tour* $(v_1, v_2, \cdots, v_{2n-1})$ for that tree (equivalently we are given an embedding of the tree in the plane and a starting vertex and edge). Then to order the vertices we will proceed as follows.

Setup: Weighting Vertex Visits

For $i = 1, 2, 3, \ldots, (2n-1)$, regard each v_i that appears in the tour $(v_1, v_2, \cdots, v_{2n-1})$ as a vertex visit.

For $i = 1, 2, 3, \ldots, (2n-1)$, assign a non-negative weight w_i to the ith visit so that the sum of weights for all visits to any fixed vertex v is one:

$$\text{For every } v \in V, \sum_{\{i|v_i=v\}} w_i = 1.$$

We call any such weight assignment a ***unit-sum weight assignment.***

An important instance of a unit-sum weight assignment assigns the same weight to all visits to the same vertex. Because each vertex v_i is visited $\deg(v_i)$ times[3], that uniform weight is exactly given by:

$$\begin{aligned} w_i &= \frac{1}{\deg(v_i)}, \text{ for } i = 1, 2, \ldots, (2n-2); \text{ and} \\ w_{2n-1} &= 0. \end{aligned}$$

Building the Sampling Interval

As we walk the Eulerian tour, we begin accumulating weights, (exactly as is done to build a weighted list for systematic sampling).

$$\begin{aligned} \text{Let } W_0 &= 0, \text{ and} \\ W_j &= W_{j-1} + w_j. \end{aligned}$$

An Illustration: Uniform Weighting. We illustrate the accumulating of weights for the uniform weighting scheme for the walk $(defegehedbabcbd)$ drawn in figure 7.

The total accumulated weights are exactly n and the total weight corresponding to each vertex of the tree is exactly one. We can assign numbers to the vertices by skipping through the weighted interval with skip interval equal to 1. This is the same as assigning an order number to a vertex each time a vertex visit takes us up to or past the next whole integer:

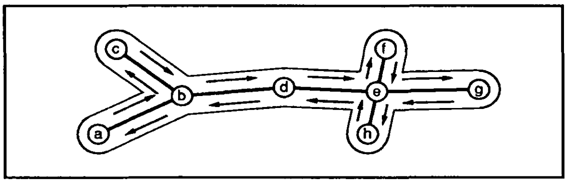
If $\lfloor W_{j-1} \rfloor < \lfloor W_j \rfloor$, then assign vertex v_j the number $\lfloor W_j \rfloor$.

Because some vertices appear in several places in our accumulated weighted interval, one may suspect that our numbering scheme may assign more than one order number to a vertex. This cannot happen.

Proofs of Correctness

The proof that the selection procedure outlined above actually produces an ordering of the vertices follows immediately from the following lemma and its first corollary.

[3]Because the Eulerian tour is cyclic, we like to count v_1 and v_{2n-1} as the same visit. We should only assign the appropriate weight to one or the other. We have chosen to assign the uniform weight to v_1.



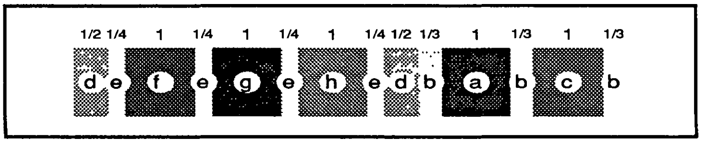


Figure 7: A Tour and its Accumulated Weights

Lemma 3 (Integral-Branch-Weights) *The fractional vertex weights accumulated between any two consecutive visits of the Eulerian tour to a multivisited vertex always add up to an integer.*

Proof: The proof of this lemma rests entirely on the observation that between two consecutive visits to any vertex v, one must depart and enter along the same edge, and an entire branch emanating from that vertex v is completely consumed by the subwalk of the Eulerian tour, as seen in lemmas 1 and 2.

In consuming an entire branch, one must visit every vertex in that branch as many times as possible, *i.e.* as many times as the degree of that vertex. Thus each vertex in the branch gets fully counted. In other words, the sum of weights for all the visits for any individual vertex during the walk of the branch is 1. And the sum of weights for *all* the visits of *all* vertices during the walk of the branch is an *integer*, equal to the number of distinct vertices in the branch. □

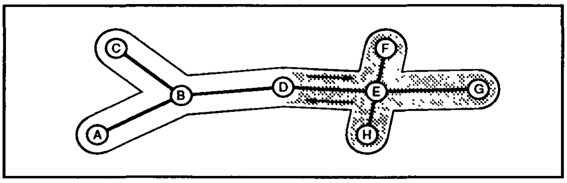


Figure 8: All Vertices of a Branch are Consumed

This lemma has two useful corollaries. To prove the first corollary we will want to talk about the fractional part of a number or an interval of numbers. Our meaning is the usual one: the fractional part of 5.35 is 0.35. The fractional part of an interval such as [17.32, 17.84) is just the set of all possible fractional values: [0.32, 0.84).

Corollary to Lemma 3.1 *Every vertex gets hit exactly once by skipping one unit at a time through the n-interval.*

Proof: Consider any vertex v of degree $= k$. Let the visits to the vertex v occur at $i_1, i_2, \cdots, i_k$. Then the visits to the vertex v will result in intervals of length

$w_{i_1}, w_{i_2}, \cdots, w_{i_k}$ being added to the cumulative interval. We want to prove that the fractional parts of the accumulated subintervals corresponding to v, namely,

$$(W_{i_1-1}, W_{i_1}], (W_{i_2-1}, W_{i_2}], \cdots, \text{ and } (W_{i_k-1}, W_{i_k}],$$

in the total interval of length n have no overlap. From lemma 3, it is clear that each successive interval, $(W_{i_j-1}, W_{i_j}]$, corresponding to a visit to v has its fractional part begin (at W_{i_j-1}) where the fractional part of the previous interval $(W_{i_{j-1}-1}, W_{i_{j-1}}]$, corresponding to a visit to v left off (at $W_{i_{j-1}}$), since an interval of integer length (*i.e.* having *no* fractional part) corresponding to all of the vertex visits of the branch consumed, has intervened.

Since the intervals $(W_{i_1-1}, W_{i_1}]$, $(W_{i_2-1}, W_{i_2}]$, $\cdots$, $(W_{i_k-1}, W_{i_k}]$, have no fractional parts overlapping and have total length equal to one, the fractional parts of values assumed in the accumulated intervals corresponding to any individual vertex must span all of the values between 0 and 1.

This last observation tells us that we can take a random start r in $[0, 1)$, take skip interval 1 once again, and we will again produce an ordering of the tree vertices. Any real number r or integral augmentation $r + m$ of r can hit at most one of the k intervals of determined by visits to v; and there is exactly one integer m_0 such that $r + m_0$ will hit one of the k intervals. □

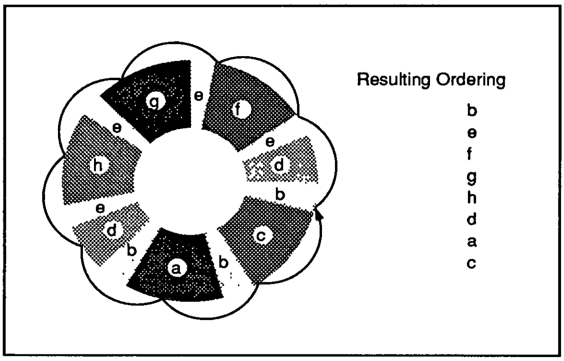


Figure 9: Cyclic Ordering with Uniform Weighting

Because the Eulerian tour is cyclic, we can make our cumulative interval cyclic and our resulting ordering cyclic as well by removing the dependence on the starting point of the tour when building our cumulative vertex-visit weight interval, as shown in figure 9. Putting all of the lemmas and corollaries together, we have the following theorem:

Theorem 1 *While making an Eulerian tour of a tree, build a separate (cyclic) interval of total length n units by assigning a non-negative weight to each vertex visit in any way so that the total weight for all visits to any individual vertex is one. Then every vertex gets hit exactly once by skipping one unit at a time through the cyclic n-interval.*

Branch-Recursion

Throughout this section we will regard the orderings generated by our ordering procedure as cyclic by making the first vertex successor to final vertex.

The next corollary follows immediately from lemma 1 and the proof of lemma 3.

Corollary to Lemma 3.2 *The collection of vertices of* **any branch** *of the tree always constitute a complete interval (i.e. appear consecutively) for any cyclic Eulerian tree-ordering.*

We will say that any cyclic vertex ordering that keeps vertices of a branch together for all branches is a ***branch-recursive ordering.*** Corollary 3.2 states that every Eulerian tree-ordering is branch recursive. It is not difficult to prove the converse using induction on branch size. We leave the proof of that theorem as an exercise.

Theorem 2 *Every branch-recursive cyclic ordering of the vertices of a tree is an Eulerian tree-ordering for some Eulerian tour of the tree and some unit-sum weight assignment to the vertex visits of that Eulerian tour.*

Branch-recursion constitutes a very strong proximity preservation property, where proximity is measured by the link-distance in the tree or graph. Branches of a tree may correspond to data clusters in cases where we have built minimum spanning trees. All quadrant-recursive orderings of a point set in the plane may be realized as orderings induced on the leaf subsets of branch-recursive orderings (*i.e.* Eulerian tree-orderings) of the quad-tree of those points.

Analysis of Complexity

An analysis of the time complexity of our tree-ordering algorithms depends on the choice of data structure with which we represent the tree. If we have a topological data structure which allows us to find the adjacent edge to any edge at any vertex in constant time, then we can order the vertices in linear time. If we need to build topology from an elementary list of vertices and edges, we can do so in time $O(n \log n)$, then proceed in linear time to complete the ordering procedure.

Space complexity is even easier to analyze. The Eulerian tour is always linear in the size of the tree. It is exactly of size $(2n - 1)$. The cumulative interval of weights that we must build is also of that size.

Enumerating Orderings

If we allow arbitrary unit-sum weighting schemes and arbitrary Eulerian tours, then we can generate all possible branch-recursive orderings. The number of branch-recursive cyclic orderings can be shown to be:

$$\prod_{v \in V} (\deg(v))!$$

as follows. Since there are $(\deg(v) - 1)!$ ways of cyclically ordering the edges incident to the vertex v, there are:

$$\prod_{v \in V} (\deg(v) - 1)!$$

possible distinct cyclic Eulerian tours. In each Eulerian tour, a vertex v may be enumerated immediately prior to any of its $\deg(v)$ branches. This results in

$$\prod_{v \in V} (\deg(v))$$

distinct cyclic orderings for each Eulerian tour. If, however, we only consider uniform unit-sum weighting schemes for fixed Eulerian tours, then we have proved [SAAL] that there are no more than:

$$LCM\{\deg(v) \mid v \in V\} \text{ distinct cyclic orderings;}$$

where *LCM* is the least common multiple. This translates into the following: If the maximum degree in the tree is 3, then there are at most 6 distinct cyclic orderings (independent of the number of vertices) for a fixed Eulerian tour. Maximum degree 4 translates into at most 12 distinct orderings; and maximum degree 5 or 6 results in at most 60 distinct cyclic orderings.

Some important trees have small maximum degree. A Euclidean Minimum Spanning Tree (EMST) of points in the plane, for example, has maximum degree 6. The EMST for points in general position has a canonical Eulerian tour as well; so the unique EMST generates at most 60 distinct cyclic orderings of points in general position in the plane, no matter how many points are in the point set!

TREE-ORDERING EDGES

Many of our results and methods for ordering vertices are equally valid for edge ordering. Theorem 1 for vertices has an exact counterpart for edges:

Theorem 3 *While making an Eulerian tour of a tree, build a separate (cyclic) interval of total length $(n-1)$ units by assigning a non-negative weight to each edge visit in any way so that the total weight for all visits to any individual edge is one. Then every edge gets hit exactly once by skipping one unit at a time through the cyclic $(n-1)$-interval.*

The proof the theorem 3 is identical to the proof of theorem 1: between consecutive vertex visits to some vertex, all (both) edge visits to any particular edge within a branch are exhausted.

Uniform Edge Weighting

A *uniform* weighting scheme for edges instead of vertices would have each edge getting weight exactly 1/2 (since every edge is visited twice in the Eulerian Tour). But giving every edge weight 1/2 amounts to nothing more than skipping every other edge in our selection procedure. So we have the following corollary to theorem 3:

Corollary 3.1 *While making an Eulerian tour of a tree, number every other edge visited. Then every edge gets exactly one number assigned to it.*

We also see immediately that:

Corollary 3.2 *Edges which are consecutively numbered using a uniform weighting scheme are never more than link distance 2 apart.*

Analysis of Complexity

As before, an analysis of the time complexity of our tree-ordering algorithms depends on the choice of data structure with which we represent the tree. With a topological data structure we can order the edges in linear time. If we need to build topology from an elementary list of vertices and edges, we can do so in time $O(n \log n)$, then proceed in linear time to complete the ordering procedure. Space complexity is once again linear for edge ordering.

Branch-Recursion

As with vertices, every Eulerian tree-order of edges in branch-recursive in the same sense:

Corollary 3.3 *The collection of edges of* **any branch** *of the tree always constitute a complete interval (i.e. appear consecutively) for any cyclic Eulerian tree-ordering of edges.*

And, conversely,

Theorem 4 *Every branch-recursive cyclic ordering of the edges of a tree is an Eulerian tree-ordering of edges for some Eulerian tour of the tree and some unit-sum weight assignment to the edge visits of that Eulerian tour.*

Enumerating Edge Orders

As with vertex orders, the number of branch-recursive edge orders is equal to the number of Eulerian tours times the number of distinct edge orders for every fixed Eulerian tour. Since each edge may be weighted so that it gets enumerated either on its first visit in the Eulerian tour or on its second visit, then as long as these two visits are not adjacent in the Eulerian tour, they will produce different orderings. Two edge visits to the same edge are adjacent in an Eulerian tour if and only if the edge is incident to a leaf vertex. A tree with more than two edges and ℓ leaf vertices has exactly $n - \ell - 1$ non-leaf edges. Thus for a fixed Eulerian tour and an arbitrary unit-sum edge weighting scheme there are $2^{n-\ell-1}$ possible orderings. The number of branch-recursive edge orderings is, therefore:

$$2^{n-\ell-1} \cdot \prod_{v \in V} (\deg(v) - 1)!$$

Enumerating uniform-weight edge orders for a fixed Eulerian tour is even more trivial than enumerating uniform vertex orders. There are exactly two uniform-weight edge orders if the tree has 3 or more edges.

ORDERING SPATIAL OBJECTS

In this section we will adapt our tree-ordering techniques to order spatial objects. Our approach in every case will be to convert the ordering problem to a tree-ordering problem, then solve the tree-ordering problem by a uniform-weighting of vertices or edges, as appropriate.

Ordering Points in the Plane

Suppose that we want to assign an ordering to a set of points in the plane. We know how to order vertices of a tree. So we may convert the points into vertices by building a tree (adding edges); and one natural tree to build is a Euclidean minimum spanning tree (EMST). The EMST is unique if the points are in general position or if no two interpoint distances are equal. So the steps needed to convert the problem of ordering points in space to one of ordering tree vertices are:

1. Build Euclidean minimum spanning tree.

2. Walk Eulerian tour, tree-ordering vertices.

We can build a Euclidean minimum spanning tree in time $O(n \log n)$ [AHO2], sorting the edges at each vertex in clockwise order as they are inserted. The planar embedding of the tree gives us the geometric version of the Eulerian tour for free (*i.e.* the usual clockwise ordering of edges around a vertex). We can then walk the Eulerian tour and order the vertices in $O(n)$ additional time.

A Cluster Sampling Application. Cluster sampling is a survey sampling strategy of selecting small groups (clusters) of neighboring points instead of selecting individual points randomly distributed. Within-cluster correlation may reduce the efficiency of such a strategy from a pure sampling viewpoint, but that consideration is often outweighed by the economic impact of reduced travel costs for interviewers.

A serious limitation to successfully selecting clusters from lists, however, is the fact that proximity in the list does not guarantee proximity on the ground. Selection of points from a list that has been ordered by performing our uniform-weight tree-ordering algorithm on a EMST of the points will guarantee very strong proximity correspondence. The following theorem holds:

Theorem 5 *Order points in the plane by building their EMST and applying the uniform-weight vertex tree-ordering algorithm. Then two consecutive points in the order have a maximum link distance of six and an average link distance of less than two.*

Proof: The degree of any vertex in a EMST is less than or equal to six. Thus the uniform-weight tree-ordering algorithm accumulates a weight of at least 1/6 with each vertex visit. Moreover, in any tree, the average degree is $\frac{2n-2}{n}$.

Ordering Points in Higher Dimensional Spaces

To apply the methods of the section on points in the plane to points in higher dimensions, we must first address two issues: (1) building a EMST in higher dimensions, and (2) defining an Eulerian tour in higher dimensions.

There are straightforward $O(n^2)$ time algorithms for building a EMST in higher dimensions [AHO1]. Some exact algorithms are known with complexity slightly sub-quadratic [YAO].

Building an Eulerian tour in higher dimensions is not so straightforward. It requires establishing a cyclic order of edges about every vertex. One possibility is to project the edges onto some two-dimensional subspace, then order the projection of the edges clockwise on that plane. Another more canonical approach, suggested by Herbert Edelsbrunner [EDE2], is to map the edge configuration about a vertex onto points on the surface of a sphere of dimension one less than the space of the EMST, then apply the ordering scheme to those points on the sphere recursively (*i.e.* build their EMST and order them in a space of smaller dimension).

In any case, if all we require is some ordering of the edges around each vertex, we can find one in $O(n \log n)$ time. We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Build EMST.

2. Cyclically order edges at each vertex.

3. Walk Eulerian tour, tree-ordering vertices.

A Sample Stratification Application. Sample stratification is a partitioning of the universe into groups which are similar across several characteristics. The characteristics should be in some sense comparable (dealing with relative incomparability is sometimes known as the Scaling Problem). Stratification is often accomplished by treating the observations as n-tuples of the n characteristics (*i.e.* as points in n-space) and finding a hyperplane or collection of hyperplanes that optimize separation of the points across the half-spaces or n-cells created. A more straightforward approach to stratification (and one that would be computationally much simpler) might be to partition a EMST of the points into branches of greatest separation. With branch-recursive ordering methods, this operation boils down to list splitting! We at the Bureau of the Census will be comparing results of using tree-ordering methods to the standard more complex stratification algorithms.

Ordering Vertices of any Graph

If we are only concerned with ordering the vertices of a graph, we may think of the graph as a tree with too many edges. So we throw away the least useful edges until we have whittled the graph down to a tree. If the edges have costs associated with them, we may wish to minimize the cost of the resulting tree, for example. We know exactly how many edges to throw away. We will discard an edge as long as it does not disconnect the graph and we still have $(n-1)$ edges left. We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Build a (minimum) spanning tree.

2. Cyclically order edges at each vertex.

3. Walk Eulerian tour, tree-ordering vertices.

Ordering Edges of any Graph

In the section on ordering vertices in a graph, we regarded our graph as having too many edges; and we threw some away. To order the edges of our graphs, we regard our graph as having too few vertices to be a tree; and we add vertices by splitting the vertices of the graph and creating more vertices with the same number of edges. Once again we use our knowledge of the edge/vertex relationship in a tree to know when to stop splitting vertices. We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Split vertices.

2. Cyclically order edges at each vertex.

3. Walk Eulerian tour, tree-ordering edges.

We must next order the tree edges about each split vertex. Then the tree edges may be assigned a cyclic order based on selecting alternate hits from an Eulerian tour of the corresponding edges of the derived tree.

Since we can certainly split vertices in $O(n \log n)$ time using sorting and a plane sweep operation, and also order edges about each vertex in some arbitrary fashion in the same time complexity, we can accomplish the following ordering for the edges of any connected graph efficiently:

Corollary 5.1 *One may find a cyclic ordering for the edges of any connected graph in $O(n \log n)$ time so that any two edges which are consecutive in the cyclic ordering never have link distance greater than two in the graph.*

Ordering Line Segments in Two-Dimensional Networks

This is just the graph-edge ordering problem, but with fewer decisions to make because the Eulerian tour is given by the geometry. The word *network* will also imply that the topological information of the graph permits linear-time generation of the ordering. The steps for converting a connected-network edge-ordering problem to a tree-edge-ordering problem are:

1. Split vertices.
2. Walk Eulerian tour, tree-ordering edges.

Ordering Line Segments Of Networks in Higher Dimensions

The difference between this section and the section on 2-D networks lies in establishing a cyclic ordering of edges about each vertex. There may be such a structure implicitly or explicitly embedded in the topological structure of the network. We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Split vertices.
2. Cyclically order edges at each vertex.
3. Walk Eulerian tour, tree-ordering edges.

Ordering Regions in the Plane

There is planar graph dual to every graph or pseudograph in the plane that is itself a pseudograph. Every region of the plane corresponds to a vertex in the new pseudograph; and two vertices in the new pseudograph are adjacent (share an edge) if and only if the regions shared a face or common side. This dual is called the *adjacency pseudograph*; and to reduce a pseudograph to a tree on the same vertex set, the procedure is the same as with a graph—you throw away edges.

We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Build adjacency pseudograph.
2. Find MST.
3. Walk Eulerian tour, tree-ordering vertices.

Application to Block Numbering. Consider the problem of numbering regions of a map in such a way that consecutively numbered regions are adjacent. It is well known that not every arrangement of blocks can be so numbered. In fact, when formulated as a problem in the adjacency graph, block numbering is nothing more or less than the problem of finding a Hamiltonian path for the adjacency graph (*i.e.* a path that passes through each vertex exactly once). Even the problem of merely deciding whether such a path exists for an arbitrary planar graph is NP-complete.

By throwing away edges so as to minimize the maximum degree of vertices in the resulting pruned tree, one may guarantee that the link distance between blocks numbered consecutively is no greater than the maximum degree of the resulting pruned tree by the same argument used to prove theorem 5.

Multistage Sampling. Sampling is often done in stages. Regions may be selected; and then individual households within selected regions may be subsampled. Region clustering, the capability of selecting groups of nearby regions, is important to reduce travel and other operational costs of surveys. *Non-compact region clustering* involves the selection of nearby, but non-adjacent regions. Non-compact clustering is an attempt to gain the benefits of reduced travel costs without the negative impact of high correlation.

Ordering regions by tree-ordering a pruned version of their adjacency graph will provide a reliable means of forming non-compact region clusters.

Ordering $(n-1)$-Cells in n-Dimensional Polytopal Regions

The adjacency dual pseudograph can be constructed for higher-dimensional cell decompositions. We may split vertices to realize the edges of the adjacency pseudograph as edges of a tree, as we do in this section; or we may prune edges and keep the vertices of the adjacency graph, as we do in the next section. We summarize the steps needed to convert the problem to a tree-ordering problem.

1. Build adjacency pseudograph.
2. Split vertices.
3. Cyclically order edges at each vertex.
4. Walk Eulerian tour, tree-ordering edges.

Ordering n-Cells in n-Dimensional Cell Decompositions

We summarize the steps needed to convert the n-cell ordering problem to a tree-ordering problem.

1. Build adjacency graph.
2. Find MST.
3. Cyclically order edges at each vertex.
4. Walk Eulerian tour, tree-ordering vertices.

CONCLUSIONS AND FOLLOW-UP

This introduction to branch-recursive orderings does not include empirical evaluations of the performance of those orderings. There was neither time to conduct those evaluations nor space to include them in this restricted paper. However, the principal reason for not assessing the performance empirically is that it is evident that these orderings will not do very well for the usual tasks of image analysis, range search, and nearest-neighbor-finding as studied in the recent comparative paper by Abel and Mark [ABEL]. Objects which are adjacent in the branch-recursive ordering are fairly close in space; however, objects which are adjacent in space may be rather distant in the branch-recursive ordering. And there is no easy way to predict how distant or when discontinuities will occur with general branch-recursive orderings, as is the case with the more common quadrant-recursive orderings. The somewhat unorthodox nearness properties that are described in this paper should, nevertheless, prove very useful for sampling activities and analysis related to those activities.

The fact that many spatial entities can be realized as or identified with vertices or edges of trees or graphs makes our results widely applicable. The following example illustrates both strengths and weaknesses of our methods. Consider the two tasks of (1) finding a cyclic ordering for n points all lying on a straight line, and (2) finding a cyclic ordering for n points all lying on a circle. The reason for considering the two tasks simultaneously is that their Euclidean Minimum Spanning Trees are topologically the same: they are both linear trees, as illustrated in figure 10.

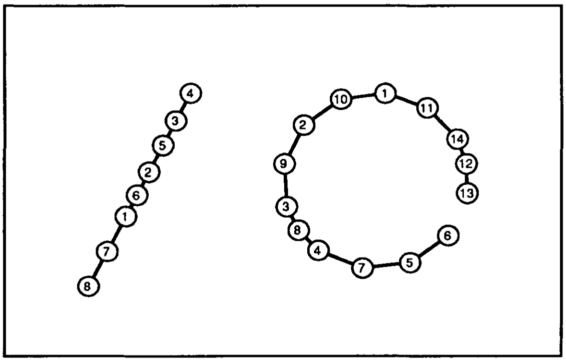


Figure 10: Cyclic Ordering of Collinear and Co-circular Points

The uniform weighting strategy will cause us to skip every other point in our numbering scheme (except at the ends of our linear tree). For the collinear points, this is clearly optimal in the following sense: This strategy minimizes the maximum distance between neighbors (*i.e.* adjacent elements in the *cyclic* numbering scheme). On the other hand, for the co-circular points, the uniform weighting strategy may produce a distance nearly double that of the optimal numbering strategy in terms of minimizing the maximum distance between neighbors.

What this example illustrates is that we necessarily lose some shape information when we embed our data in a tree and use only the topological structure of the tree from that point on. What the example also illustrates is that we may in some cases get optimal performance for cyclic orderings.

REFERENCES

[ABEL] Abel, David J.,and David M. Mark, 1990, *A Comparative Analysis of Some Two-Dimensional Orderings*, **International Journal of Geographical Information Systems**, 4(1), 21-31.

[**AHO1**] Aho, Alfred, John Hopcroft, and Jeffrey Ullman, 1974, **The Design and Analysis of Computer Algorithms**, Addison-Wesley, Reading, MA.

[**AHO2**] Aho, Alfred, John Hopcroft, and Jeffrey Ullman, 1985, **Data Structures and Algorithms**, Addison-Wesley, Reading, MA.

[**EDE1**] Edelsbrunner, Herbert, 1987, **Algorithms in Combinatorial Geometry**, Springer-Verlag, New York.

[**EDE2**] Edelsbrunner, Herbert, 1990, personal communication.

[**FAL1**] Faloutsos, Christos and Yi Rong, 1989, *Spatial Access Methods Using Fractals: Algorithms and Performance Evaluation*, University of Maryland Computer Science Technical Report Series, UMIACS-TR-89-31, CS-TR-2214.

[**FAL2**] Faloutsos, Christos and Shari Roseman, 1989, *Fractals for Secondary Key Retrieval*, University of Maryland Computer Science Technical Report Series, UMIACS-TR-89-47, CS-TR-2242.

[**GARE**] Garey, Michael R., and David S. Johnson, 1979, **Computers and Intractability, A Guide to the Theory of NP-Completeness**, W. H. Freeman, New York.

[**HARA**] Harary, Frank, 1969, **Graph Theory**, Addison-Wesley, Reading, MA.

[**KISH**] Kish, Leslie, 1965, **Survey Sampling**, John Wiley, New York.

[**MARK**] Mark, David M., 1990, *Neighbor-based Properties of Some Orderings of Two-Dimensional Space*, **Geographical Analysis**, April, 22(2), 145-157.

[**PREP**] Preparata, Franco, and Michael Shamos, 1985, **Computational Geometry, An Introduction**, Springer-Verlag, New York.

[**SAAL**] Saalfeld, Alan, 1990, *Canonical Cyclic Orders for Points in the Plane*, submitted to **Journal of Computational Geometry: Theory and Applications**, Elsevier.

[**WOLT**] Wolter, Kirk, and Rachel Harter, 1989, *Sample Maintenance Based on Peano Keys*, presented at Statistics Canada Symposium on Analysis of Data in Time, Ottawa, Canada.

[**YAO**] Yao, Andrew Chi-Chih, 1982, *On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems*, **SIAM Journal of Computing**, November, 11(4), 721-736.

Zenithial Orthotriangular Projection

A useful if unesthetic polyhedral map projection to a peculiar plane

Geoffrey Dutton[1]
Spatial Effects
150 Irving Street
Watertown, MA 02172 USA
qtm@cup.portal.com

Abstract

This paper describes the construction, properties and potential applications of a cartographic projection recently developed by the author, called the *Zenithial Orthotriangular (ZOT) projection of an Octahedron*. *ZOT* maps a planet to a plane by modelling it as an octahedron (a regular solid having 8 equilateral triangular facets), which is then unfolded and stretched to fit within a square. As described below, *ZOT* is developed from a regular octahedron mapped in North polar aspect, by cutting octant edges of the southern hemisphere from pole to equator, and stretching all octahedral facets to occupy eight identical right triangles (extensions to the ellipsoid are described). The North pole lies at the center of projection, while the South Pole occupies all four corners; points along map borders are mirrored across the central axes. After discussing its cartographic properties, *ZOT*'s relation to the Quaternary Triangular Mesh (*QTM*) global tessellation is explored. The use of *ZOT* is shown to facilitate recursive definition of *QTM*'s geodesic graticule of nested triangles. Computationally, this structure is handled as a quadtree, even though its elements are triangular in shape. Basic procedures for mapping geographic coordinates to *QTM* quadtree addresses via *ZOT* are presented and discussed, and suggestions given for standardizing how *QTM* tiles are addressed in *ZOT* space.

[1] The author gratefully acknowledges encouragement and support for this work from Prime Computer, Inc.

Polyhedral Maps

There is a family of maps called *polyhedral projections* that apportion regions of Earth to coincident facets of some concentric polyhedron. If the polyhedron is one of the five platonic solids, these facets will be either square, pentagonal or most likely, triangular, and all the same size and shape. While these figures may be torn apart and unfolded in a number of ways, no regular polyhedron beyond the tetrahedron can be unfolded to lie on the plane in a maximally compact way; there will always be concavities whatever arrangement of facets is used. As a consequence, polyhedral maps tend to have convoluted, lobed shapes, rather than fitting neatly into a rectangle, as do most projections. This apparently frustrates cartographers, who often seem to feel that polyhedral projections involve excessively complicated computational procedures. This is only partly true: however odd and enigmatic such constructions may be, they are at least *regular* and *enumerable.*

Mapping regions of the Earth to facets of a polyhedron can involve any of a number of map projections, the most natural of which is the *gnomic.* This is one of the few projections in which all coordinates relate to a single point of reference (the center of the planet). Although gnomic projections are not suitable for large areas, their distortions are quite minor when limited to the facets of enclosing polyhedra. Most azimuthal projections (such as the stereographic) require multiple reference points in order to portray the entire globe. This paper describes an azimuthal mapping of of an *octahedron* to a *square* in North polar aspect.

Projective Properties

The *ZOT* projection is *zenithial* (azimuthal) because meridians remain straight and of constant radial spacing; longitudes may be measured directly with a protractor. There is, however, more than one azimuthal origin, as longitudes are only true within a hemisphere. As the South pole is separated into four locations, meridians in the southern hemisphere originate at each of the four corners of the projection. *ZOT* also has the *equidistant* property; distance between parallels is constant throughout the map. The projection has been named *orthotriangular* because it maps spherical triangles to right triangles in its domain. These properties are evident in the world map in ***Figure 1****. ZOT* is also *doubly periodic;* that is, it may be repeatedly tiled in two directions to fill a plane, as ***Figure 2*** illustrates.

Zenithial Orthotriangular Projection

North polar aspect, shown with 15º graticule

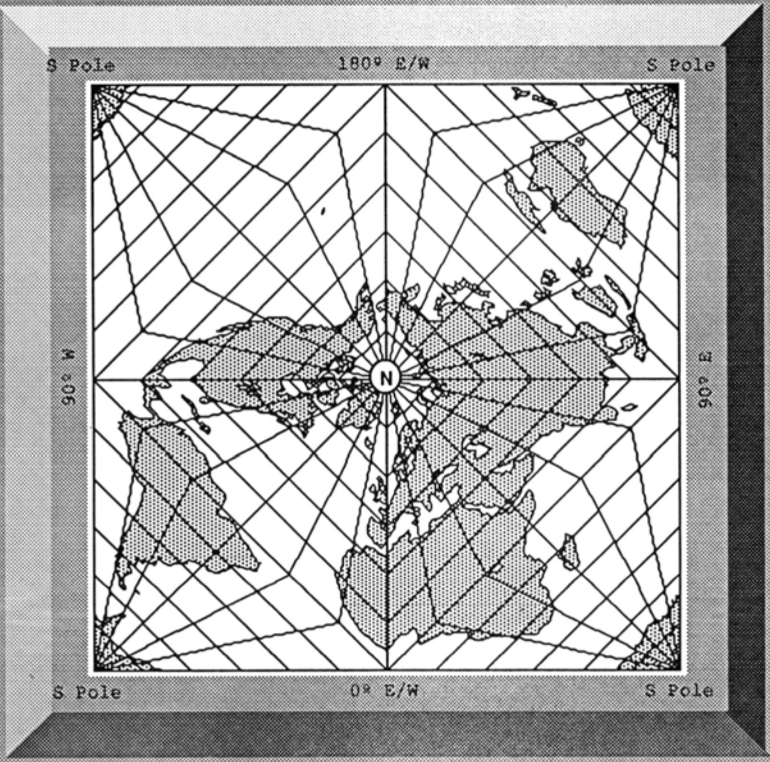


Figure 1

Zenithial Orthotriangular Wallpaper

North and South polar aspect, shown with 15º graticule

ZOT is doubly periodic (repeating in X and Y)

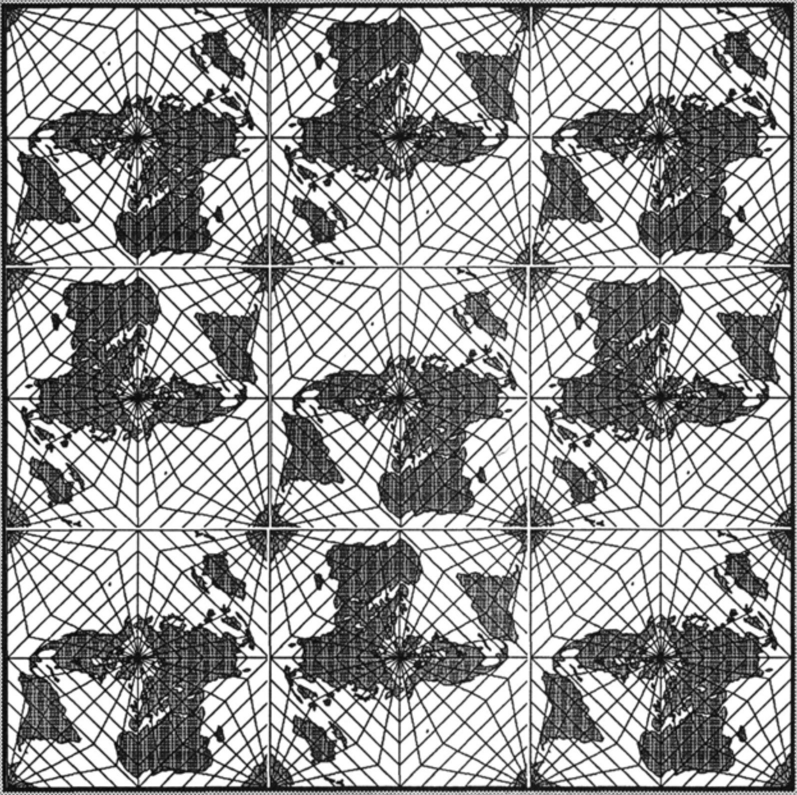


Figure 2

ZOT is neither *equal-area* nor *conformal.* Along parallels, map scale varies inversely with latitude, with the error factor growing from unity at the pole to √3 at the equator. This occurs because the equilateral facets of the octahedron are mapped to right triangles, causing their equatorial bases to expand. Along any given meridian, map scale is constant. However, the scale varies linearly from one meridian to the next, from unity (at 45, 135, -135 and -45 degrees) to √2 (at 0, 90, 180 and -90 degrees longitude), cycling four times around the equator. In general, there is no scale error at the poles, a small amount in the vicinity of the 8 octa face centers and more near their edges, being greatest along the four equatorial edges, and increasing toward the four equatorial vertices (which occupy the midpoints of *ZOT* map margins).

Despite this variability, all meridians map to straight lines which flex at the equator, and parallels to straight lines which flex at each 90th meridian, due to the piecewise continuous (polyhedral) nature of the projection. In most polar azimuthal projections, parallels map to circles or ellipses. In the orthotriangular projection, they map to diamonds (squares). This derives from the distance metric ("Manhattan") employed, and reflects the fact that the projection maps a sphere to the planar facets of an octahedron. This rectalinearity and modularity makes the projection very easy to compute, as it permits geographic coordinates to be mapped to the plane using linear equations, without recourse to trigonometric formulæ, square roots or, under restricted conditions, real arithmetic.

One obvious, even disturbing, property of ZOT is the 90º change in direction of parallels at every 90th meridian. This causes strange distortions in the shapes in all major land masses other than South America and Australia. Likewise, the flexing of meridians at the equator distorts Africa and South America. The former effect can be minimized by offsetting meridional octant edges roughly 25º to the West, which bisects land masses at more natural locations. The latter effect cannot be mitigated, as the equator cannot be shifted in any useful way. For computational purposes ZOT's orientation is rather immaterial, but should be standardized (see suggestion below).

Computing ZOT Coordinates

When a point is to be projected, its colatitude is multiplied by the map scale; the product is multiplied by the point's longitudinal displacement from the left edge of the octant and divided by $\pi/2$. The result is either an x or y offset from the pole's location, depending on the octant within which the point lies. We compute the other offset by subtracting the first one from the scaled colatitude; this fully determines the point's x,y location on the map.*The procedure's simplicity derives from using "city block" distances (Manhattan Metric), in which distance between points is the sum of x and y displacements, instead of Pythagorean distances.* In other words, all points along a given *ZOT* latitude are equidistant from the pole closest to them (the sum of x and y is constant and proportional to colatitude). The locus of all points along a given latitude is a straight line cutting through the octant at 45º (parallel to its equatorial base); a given distance traversed along a parallel has a size proportional to longitude, another simple linear function. The *ZOT* projection for the North polar aspect may be derived as follows:

Derivation of ZOT x,y coordinates from geographic Locations

Given:

```
double Plat     := Latitude being projected  -- In Radians
double Plon     := Longitude being projected -- In Radians
double Diam     := Map diameter        -- Cm, inches or other linear unit
double S        := Diam / π            -- Absolute scale factor
double P2       := π / 2               -- Constant for right angle
```

Parameters:[2]

```
int    OCT      := Octant occupied by point -- 1–4 in N, 5–8 in S Hemi
double R[1]     := X-coordinate Scale factor -- Sign only varies by octant
double R[2]     := Y-coordinate Scale factor -- Sign only varies by octant
double C[1]     := X-coord origin for octant -- Center, left or right side
double C[2]     := Y-coord origin for octant -- May be center, top or bottom
int    FLOPS[8] := {1, 1, -1, -1, -1, 1, 1, -1}  -- Meridional edge orientations
```

Set up Octant:

```
int    ORG      = (P2 - Plat) div P2       -- Map origin (0 = center, 1 = corner)
int    OCT      = (ORG + 1) * (Plon div P2)   -- Octant occupied (1–8)
int    X1       = 2 - ((OCT + ORG - 1) mod 2) -- 1 if Lat maps to X, 2 if to Y
int    X2       = 3 - X1                   -- 2 if Latitude maps to X, 1 if to Y
int    HS       = 1 - (2 * ORG)            -- Hemisphere Sign (1 in N, -1 in S)
double R[X1]    = S * FLOPS[OCT]           -- X or Y factor (-R left, +R right)
double R[X2]    = - S * HS * FLOPS[9 - OCT] -- Y or X factor (-R top, +R bot)
double C[X1]    = - ORG * R[X1]            -- X or Y Center (Zero in N hemi)
double C[X2]    = - ORG * R[X2]            -- Y or X Center (Zero in N hemi)
```

[2] The parameters and variables in this algorithm are typed according to their basic cardinalities. Certain *int* parameters are also used in floating point expressions (performed in *double* precision, we presume); ints to can be converted to real as one's programming environment may require.

Project Point:

```
double  CLP    = P2 - (HS * Plat)          -- Absolute Colatitude of point
double  OLP    = CLP * (Plon mod P2) / P2  -- Long offset (prop. to Colat)
double  PX     = R[X1] * abs(CLP - OLP)    -- Relative X or Y offset
double  PY     = R[X2] * OLP               -- Relative Y or X offset
PX             = C[X1] + PX                -- Projected X Coordinate
PY             = C[X2] + PY                -- Projected Y Coordinate
```

After initial octant setup calculations (which involve computing only 9 numbers and, in most cases, need be done but a few times for a given set of coordinates), the above algorithm uses 4 additions, 4 multiplications, 1 division and 2 rational function calls to map one point from the sphere to the plane. In situations where the octant points occupy changes frequently, setup can be table-driven based on an octant number, just as table FLOPS provides signs of scale factors and axis origins.

Note that while the above algorithm assumes a spherical Earth, its principle can also be applied to ellipsoids, at the expense of some additional arithmetic. Table FLOPS represents lengths (unity) and orientations (sign) of edges of an octahedron enclosing the planet. Were this object to have non-uniform semiaxes, the entries in FLOPS would have values differing slightly from unity; this data could be used to anchor the projection to any specified ellipsoid. In the spherical case, one computes Y coordinates along a line having its intercept at *Plat* and a slope of unity, scaling X from *Plon*; for ellipsoids, the procedure involves slopes differing slightly from unity, but is otherwise handled identically to those more complex cases.

Related Antecedents

The *ZOT* is not the first world projection into a square domain having double periodicity, nor is it the first to exploit the geometry of the octahedron. It apparently is the first to employ a *Manhattan* distance metric, and one of the few which can be constructed without trigonometric functions (such as the *Peters* or *equirectangular*). One of its more interesting predecessors is the *Quincuncial* projection, developed in the 1870's by Charles Sanders Peirce. Based on elliptic integrals, this remarkable and elegant construction is *conformal* and *doubly periodic*.[3] Despite its obvious octahedral symmetry, Peirce apparently never related his projection to polyhedra. Although widely appreciated, it fell into disuse, although the Coast and Geodetic Survey used it in a 1947 world navigation map (Eisele, 1963).

[3] *Quincunx* is a Latin word meaning "arrangement of five things." Peirce's *Quincuncial* projection is just that, as it places the South pole at the corners of a square and the North pole at its center.

Also related to *ZOT* is Cahill's *Butterfly* projection (Fisher and Miller, 1944), an interrupted conformal projection of the globe onto eight triangular facets arranged in a butterfly-like shape. In each of its octants, the equator and central meridian are straight and all other meridians and parallels bow outward. As a result, assembly of the *Butterfly* results in a lumpy shape somewhere in between an octahedron and or a sphere. Also, indexing map locations is complicated both by the mathematics required for the *Butterfly* projection and the arrangement of its facets.

Buckminster Fuller's *Dymaxion* projection dates from the 1940's and seems to have undergone a metamorphosis from an initial cuboctahedron basis[4] to the icosahedral form of the version currently marketed (Life, 1943; Fisher and Miller, 1944; Fuller, 1982). Fuller's and Cahill's motivations seem to have been similar in producing these projections; to minimize scale errors and to exploit polyhedral geometry to produce a globe that can be folded from a single sheet of paper. Fuller was keen on using his projection to convey thematic data about "Spaceship Earth", (he envisioned a large Dymaxion geodesic globe studded with computer-controlled miniature lamps to depict global statistical data, but seems never to have done this). Most versions of the *Dymaxion* employ gnomic projections.

The *"polygnomic"* world projection onto an icosahedron may have first been realized by Fisher (Fisher, 1943), even though Fuller enjoyed taking credit for it. Indeed, the idea (if not its execution) can be traced back to the work of Albrecht Dürer in the sixteenth century (Fisher and Miller, 1943, p. 92). This invention suited Fuller's purposes perfectly, as it represents chords of great circles with straight lines, like the struts of one of his geodesic domes. *ZOT*, however, is *not polygnomic*; it is oriented to the poles, not to the center of the Earth. Consequently, most great circles are not straight lines in *ZOT* space (but the equator and all meridians are).

[4] A cuboctahedron is a 14-sided polyhedron having 8 triangular and 6 square facets. Unlike the five regular polyhedra, the facets are tangent to two concentric spheres, complicating construction or calculation of features that cross facet edges.

Error Adjustment

Nearly any area or distance measured from an *ZOT* projection will be incorrect by as much as a factor of two. As it is almost as simple to calculate the scale error at any point as it is to compute coordinates, and only slightly harder to derive the error involved when distances between points or polygonal areas are computed (with cases involving more than one octant presenting the most complexity). This means that size and distance calculations may be corrected as required; the greater the precision, the greater the cost. Tables can be developed to facilitate such corrections.

Polyhedral Addressing

ZOT is not esthetically pleasing, especially in comparison to the sweeping curves of Peirce's *Quincuncial. ZOT* generates angular discontinuities at octant boundaries, violating a number of cartographic precepts. No claim is made for it as an optimal visual matrix for presenting global spatial data. Still, *ZOT projection may have considerable computational utility* when applied to tessellated polyhedra embedded in a well-defined spherical manifold, as the following section explains.

The best uses for *ZOT* may be those which capitalize on its computational simplicity. In particular, there is a strong affinity between *ZOT* and the geometry of the Quaternary Triangular Mesh (*QTM*) global location coding model (Dutton, 1989; Goodchild and Yang, 1989). ***Figure 3*** and ***Figure 4*** illustrate how *QTM*'s recursive subdivision of octahedral facets into four tiles each is mapped to a completely regular mesh of right triangles when projected via *ZOT*. This mesh densifies in the same manner as a rectangular quadtree does, but also includes diagonal elements (parallels of latitude). Note how each triangle's edges split in half, and how its hypotenuse follows a particular latitude. This may be exploited to derive *QTM* facet addresses from latitude and longitude, as ***Figure 5*** shows.

The arithmetic used in this procedure consists of testing sums and differences of x and y displacements against one parameter (*s*/2 in fig. 5) that is constant for all *QTM* tiles at a given level of detail. In addition, the algorithm needs to know the "basis number" of each node (vertex) in the *QTM* network in order to assign a *QTM* ID to every tile in the hierarchy; each vertex is identified with a 1-node, 2-node or 3-node (its basis number), and all higher-level nodes at a particular location continue to manifest its original basis number. This digit is common to all four *QTM* cells surrounding each octa vertex, and all six cells that surround the nodes that appear in subsequent subdivisions. Central (0) cells are associated

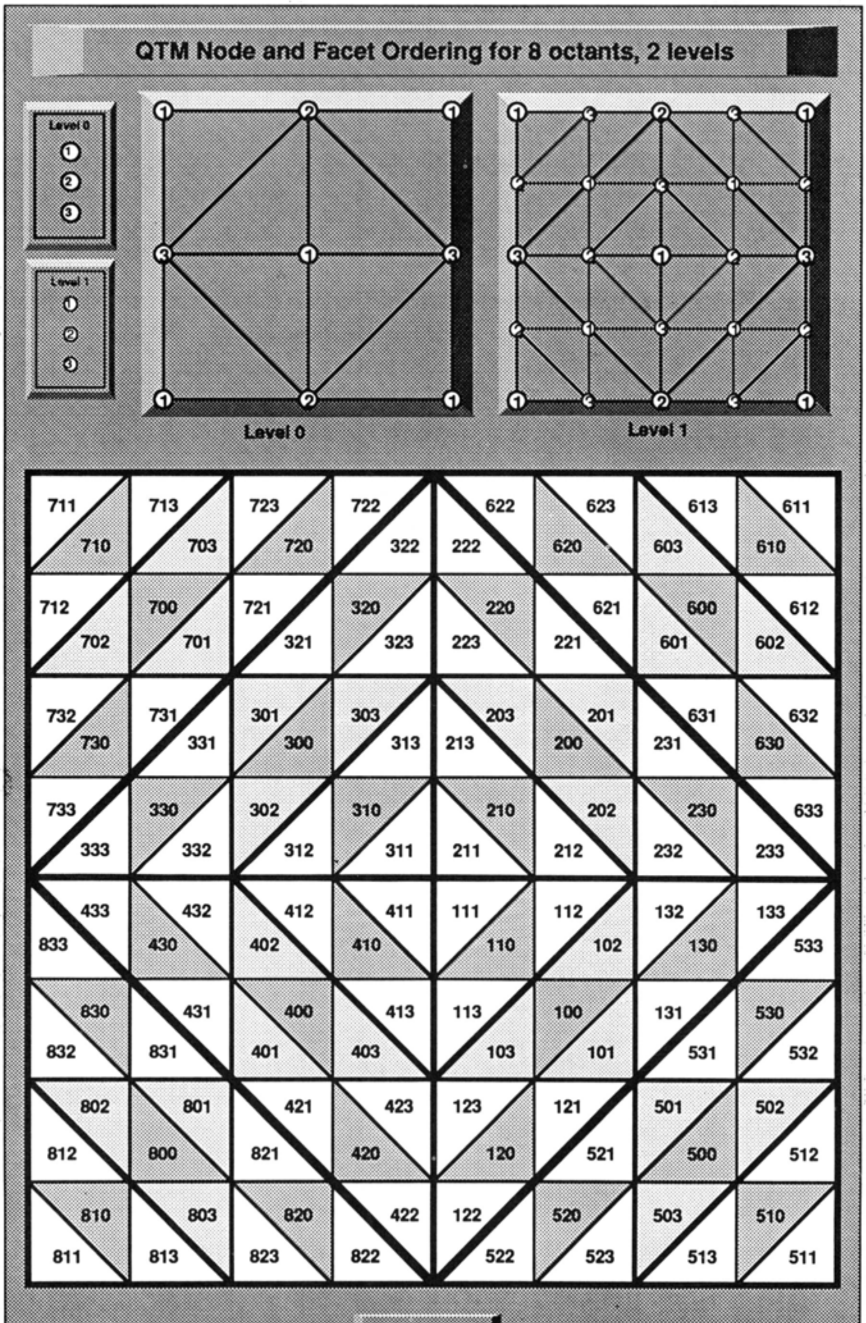
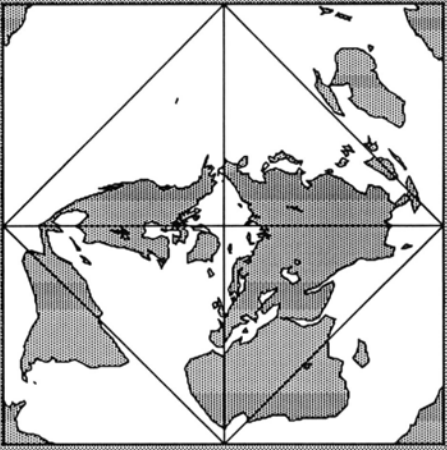
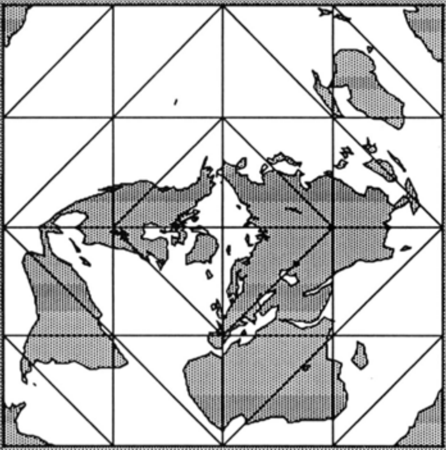
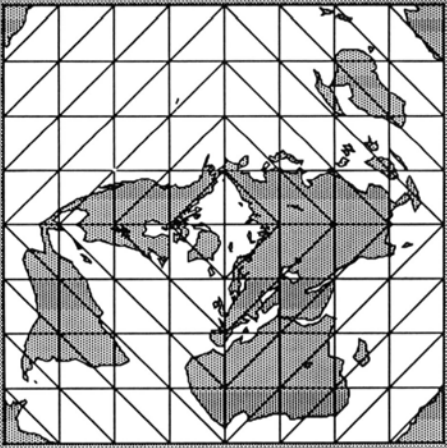

QTM Node and Facet Ordering for 8 octants, 2 levels
Level 0
Level 1
Level 0
Level 1
711 710 713 703 723 720 722 322 622 222 623 620 613 603 611 610
712 702 700 701 721 321 320 323 220 223 621 221 600 601 612 602
732 730 731 331 301 300 303 313 203 213 201 200 631 231 632 630
733 333 330 332 302 312 310 311 210 211 202 212 230 232 633 233
433 833 432 430 412 402 411 410 111 110 112 102 132 130 133 533
830 832 431 831 400 401 413 403 113 103 100 101 131 531 530 532
802 812 801 800 421 821 423 420 123 120 121 521 501 500 502 512
810 811 803 813 820 823 422 822 122 522 520 523 503 513 510 511

Figure 3

Development of QTM grid in ZOT space


(a) Level 0 (octahedron facets)


(b) Level 1 (32 facets)


(c) Level 2 (128 facets)

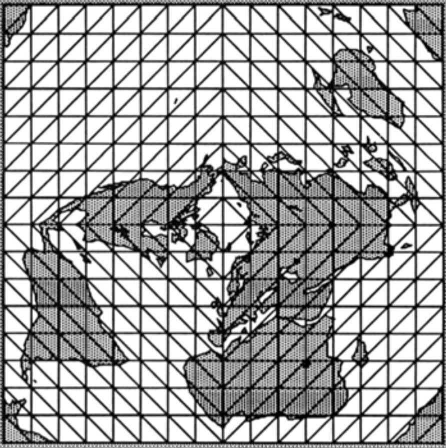

(d) Level 3 (512 facets)

Figure 4

with no node, but their vertices (and subsequent cells that surround them) themselves have node identifiers.

To map geographic coordinates to *QTM* identifiers, an additional procedure is therefore needed: one which identifies the "pole node"[5] (the right-angled vertex) of each *QTM* cell, and also assigns correct basis numbers to all three nodes (pole nodes can have IDs of 1, 2 or 3). This is a property not of the *ZOT* projection itself, but of the sequencing of 1- 2- and 3-cells at each level in the tessellation, which may be done as specified here, as Goodchild and Yang (1989) describe,[6] or in some other way. Another aspect of navigating *QTM* which must be parametrized is the geometric orientation of principle axes with respect to the pole node of each facet, which can be either of two arrangements per octant, one for ID's 1, 2 and 3, the other involving ID's of zero. When a point occupies a central (0) facet, the facet's orientation inverts, rotating 180 degrees. This new arrangement persists until a zero ID recurs, at which point the facet shrinks by 50 percent and flips into the other orientation. The rule is: *all facets within a given octant share its orientation unless their QTM codes contain an odd number of zeros; in such cases the current x and y scale factors interchange and change sign.*

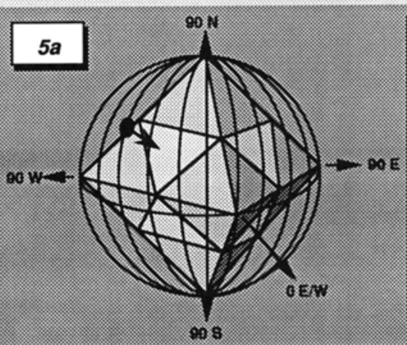
When a 0-tile comes into being, its pole node is a reflection of, and has the same ID as its parent *QTM* facet's pole node. What had been half of its parent's x-extent becomes the 0-tile's y-extent, and *vice versa*. In cases where the child tile is in the triangle dominated by the parent's pole node, its ID will be the same as its parent's. In either of the remaining two (nonzero) cases, the ID of the child's pole node flips from that of the node to which it is closest to that of the other non-pole node. Once embedded in the *ZOT* plane, transitioning to certain *QTM* ID's involves horizontal displacement, while vertical movement is used to reach others (x and y in *ZOT* space; see ***Figure** 5*). Three of the six possible arrangements of nodes within an octant are enumerated in ***Table 1*** and diagramed in ***Figure 6***.

[5] This is the local origin of each facet, the vertex in the *QTM* mesh that, as projected via *ZOT*, has edges that all meet at right angles. Local *ZOT* distances are measured with respect to this origin, which moves each time a *QTM* ID assumes a new value.

[6] Goodchild and Yang number the tiles their mesh from 0 to 3 in one of two patterns that spiral out from the the central (0) tile first either North or South (1), then Southwest or Northwest (2), then East (3). While this scheme may simplify trilocation (generating tile IDs), it lacks one important property: There is no correspondence between tile ID's and vertex basis numbers; this makes it more difficult to relate tiles to the nodes they surround (their *QTM Attractors*).

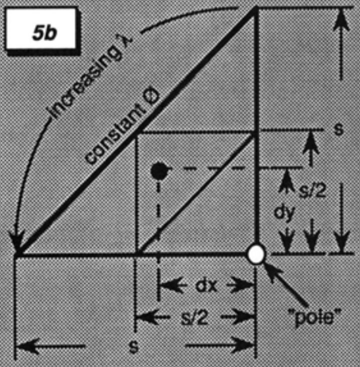
Deriving QTM Codes via ZOT Arithmetic

5a



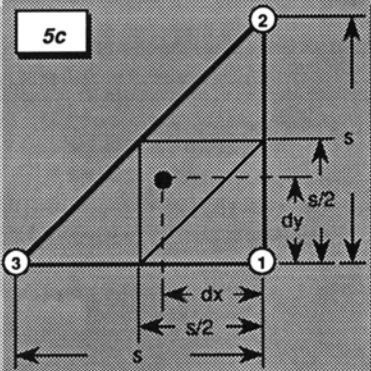
One derives QTM code digits recursively by, at each level, identifying which of four tiles encloses a point occupying latitude (Ø) and longitude (λ). This position is referenced to a local origin ("pole"), yielding ∂Ø and ∂λ (angular displacements within a QTM cell). The number returned identifies the closest QTM attractor (node).

5b

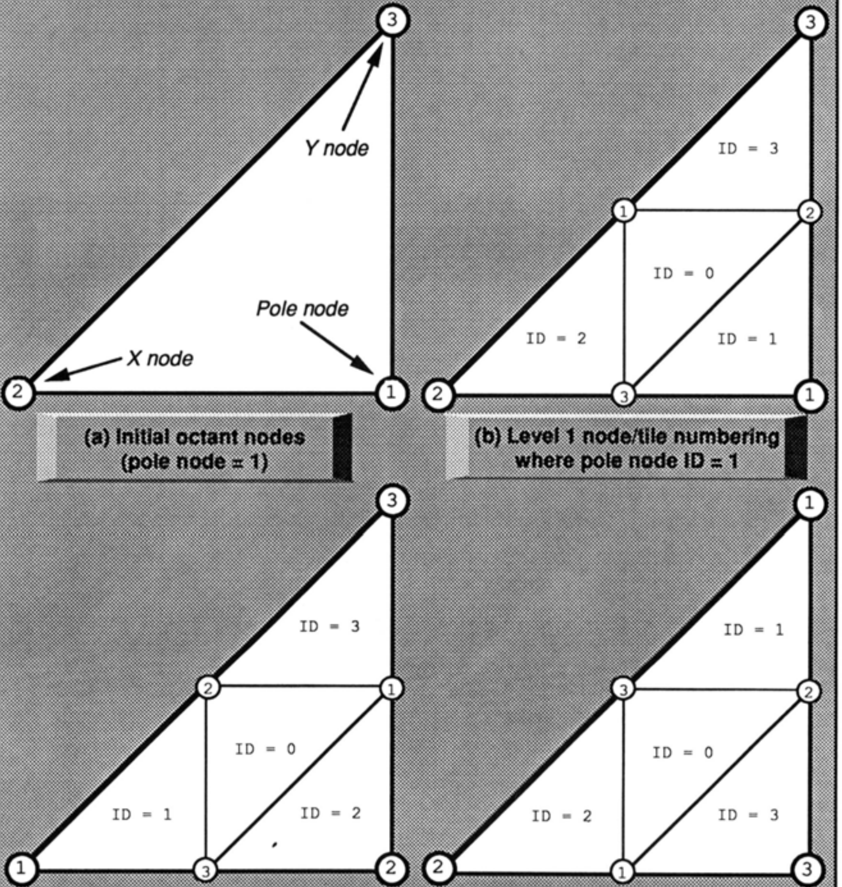


Get s; the length of triangle legs.
Get s/2; half of s.
Get dx; point x-offset from origin
Get dy; point y-offset from origin
{ s is angular; := 180 / (2 ^ level),
as measured from pole }
{ dx & dy are also angular offsets }

5c



```
s = 90.; side length in degrees
s/2 = 45.; half side length
dy = ∂Ø; latitude change from origin
dx = ∂λ - dy; other coordinate
If (dx+dy) < s/2 then return (1);
If dy > s/2 then return (2);
if dx > s/2 then return (3);
else return (0);
```



Arrangements of QTM nodes in ZOT space

(a) Initial octant nodes (pole node = 1)

(b) Level 1 node/tile numbering where pole node ID = 1

(c) Level 1 node/tile numbering where pole node ID = 2

(d) Level 1 node/tile numbering where pole node ID = 3

figure 6

Table 1

Basis numbers of nodes of children of an octa facet
(3 of 6 orientations)

OTM Tile	Pole	X-Node	Y-Node	- Figure 6
Parent	1	2	3	
0	1	2	3	
1	1	3	2	
2	3	2	1	
3	2	1	3	

OTM Tile	Pole	X-Node	Y-Node	- Figure 6c
Parent	2	1	3	
0	2	1	3	
1	3	1	2	
2	2	3	1	
3	1	2	3	

OTM Tile	Pole	X-Node	Y-Node	- Figure 6d
Parent	3	2	1	
0	3	2	1	
1	2	3	1	
2	1	2	3	
3	3	1	2	

Note how in each case, if a point lies nearest the parent's pole node, the child will have the same pole, but the x-node and the y-nodes interchange ID's.

Computational Properties

Because planar geometrics are generally much more straightforward than spherical ones, it is almost always easier to compute relations such as distances, azimuths and polygon containment on the plane rather than on the sphere. The former may involve square roots and occasional trig functions, but rarely to the degree demanded by geographic coordinates, where spherical trigonometry must be used no matter what ranges may be involved (unless approximations will suffice). Polyhedral geometry, being closed and faceted, is globally spherical but locally planar. The maximum practical extent of localities varies, both in cartesian and faceted cases, according to the projection employed (for cartesian coordinates) or the type and level of breakdown (for hierarchical polyhedral tessellations).

One essential operation that *ZOT* can facilitate is computing polyhedral facet addresses (geocodes) from geographic coordinates. Called *trilocation* (Dutton, 1984), it recursively identifies the ID's of tiles containing a given location, generating a sequence of *L* 2-bit codes, where *L* is the depth of recursion. The simplest general algorithm for trilocating a point in *QTM* determines which of four tiles it is in by comparing squared distance from the specified point to the centroids of the central *QTM* tile and each of the three outer ones to find the closest one; this requires 1 to 3 squared distance computations and comparisons per level, or O (2*L*) comparisons per point. If performed in global space, great circle distances are needed, but in the plane cartesian distances will suffice (in neither case need square roots be extracted, as we need only order distances, not measure their absolute magnitudes). *In ZOT space, computing a QTM ID requires only one addition, one subtraction, and one, two or three tests of inequality,* as demonstrated in ***Figure 5.***

ZOT casts trilocation into a well-defined planar geometry where triangular cells can be efficiently identified. Moreover, one may compute facet ID's to 15 levels of detail using coordinates stored as 32-bit integers (attempting greater precision would cause overflows and aliasing of IDs beyond the 15th level). Projecting candidate points from longitude and latitude into *ZOT* coordinates only involves solving several linear equations per point. *ZOT* distances order themselves the same as geodesic distances, and as just described, are much easier to compute.

Orientation Options

The *ZOT* projection has been shown in a specific orientation throughout this paper. As mentioned above, it is trivial to rotate the Prime Meridian to cross any point on the equator. This relocates four *QTM* cardinal points and all octant boundaries; one may be tempted to do so to avoid spreading areas of interest over more than one or two octants. Such schemes are always to the advantage of certain territories at the expense of others. Such suboptimizations are probably self-defeating, and in any case violate the spirit of the model: *QTM* can best identify locations on a planet if its mesh is embedded in a particular manifold (topological reference surface) in an agreed-upon way. Differently-oriented manifolds generate different *QTM* codes for the same location; this complicates spatial analysis, as codes from *QTM* model variants that do not share a common orientation are not commensurate, even when they represent identical locations.

QTM isn't very useful unless it is standardized, as are latitude and longitude. If nothing else, *QTM* is a coordinate system, designed to recursively encode (at some specified precision) locations on planets into unique triangular facets. It is therefore desirable that all *QTM* codes having a given address map to the same location on a planet, no matter who specified the address, where they came from or for what purpose. This implies that certain areas will always be inconveniently split by octant boundaries. Such situations can be handled by methods which knit facets together along octant edges, such as associating them with *QTM attractors*[7] (which as *figure 3* shows, follow the same pattern in all eight octants). Were everyone who used the framework to agree on how to orient it, all their *QTM* codes would also agree. Little additional data (mainly an ellipsoid model) is required beyond a common definition of the octahedron's orientation to the planet concerned.

Table 2 proposes a standard way to orient *QTM* to *ZOT*, used in illustrating this essay. It is defined by three parameters that relate *QTM* nodes to *ZOT* space: (1) The projection's *aspect* (North polar); (2) the longitudinal *offset*, if any, for the prime meridian (0º); (3) the cardinal *direction* from the central axis along which the prime meridian runs (-Y). If the geographic North and South poles are assigned ID's of 1, and the intersection of the equator with longitude 0º and 180º are labeled 2, the remaining two octahedral nodes (where the equator and longitudes -90º and 90º cross) therefore have ID's of 3. This fully defines the basis number of every node in the entire *QTM* hierarchy. The *ZOT* coordinates for x and y nodes are given in terms of the map radius (which is the length of octahedral edges as projected). These are either zero, or plus or minus unity.

[7] *QTM* nodes are also called *attractors* because all coordinates in the vicinity of a node alias to it, hence can be thought of as being attracted to that location. All *QTM* nodes beyond the original six octahedral vertices propagate their ID to six surrounding tiles, and all coordinates falling within those tiles are associated with the attracting node.

Table 2a

Proposed QTM Orientation Standard for Octa Vertices

(Octa vertices define 3 orthogonal axes upon which all QTM codes are based)

Latitude	Longitude	Pole	ZOT-X	ZOT-Y
90 N	(0)	1	0	0
90 S	(0)	1	±1	±1
0 N/S	0 E/W	2	0	1
0 N/S	180 E/W	2	0	-1
0 N/S	90 E	3	1	0
0 N/S	90 W	3	-1	0

Table 2b

Proposed QTM Orientation Standard for Octa Facets

(-x = left; +x right; -y up; +y down w.r.t. Pole node. Signs are descriptive only; node IDs are positive)

Octant	N/S	Pole	X-ID	Y-ID
1	N	1	3	2
2	N	1	3	-2
3	N	1	-3	-2
4	N	1	-3	2
5	S	1	-2	-3
6	S	1	-2	3
7	S	1	2	3
8	S	1	2	-3

Projected Implications

It is not foreseen that zenithial orthotriangular projection will ever be widely employed in published maps. *ZOT* is too peculiar to serve as an aid to navigation or to be used to convey thematic data (unless its double periodicity can be exploited)[8]. What it offers, however, is a computational shortcut for spatially indexing locations on a planet. This approach follows the lead of Lucas (1979), Diaz and Bell (1986) and others in attempting to define special arithmetics for tessellated spatial data in order to take advantage of properties of particular tessellations. Although the spaces in which most such arithmetics operate cannot be visualized as readily as *ZOT* space can, tessellar methods can have considerably higher computational efficiencies than standard geometric calculations.

[8] One might convey bivariate (or even trivariate) attribute data using a tiling of ZOT maps (as Figure 2 shows). For example, a thematic variate, such as population densities, could be displayed in a grid of M maps across, each column representing a different date in history (e.g., 1950, 1970 and 1990); each of N rows of the grid might display a different spatial resolution (one could display densities computed over the areas each nation, province or canton, one row for each scale).

ZOT can greatly simplify repetitive geometric operations in a quaternary triangular mesh, as we have tried to describe. *QTM* facets are optimally arrayed in *ZOT* space, and their addresses are highly tractable to compute. Deriving *QTM* ID's from geographic coordinates via *ZOT* is algorithmically inexpensive, growing more or less as $\mathbf{O}(L \log L)$. So, *ZOT* may prove to be a useful cartographic abstraction, at least to the extent that *QTM* is a felicitous framework for spatial data.

References

Diaz, B. and S. Bell, 1986; *Spatial data processing using tesseral methods,* **Proc. Tesseral Workshops** 1 and 2 (1984 and 1986). U. of Reading, UK: NERC Unit for Thematic Information Systems.

Dutton, G.H., 1984; *Geodesic modelling of planetary relief,* **Cartographica**. Monograph 32-33, vol. 21, no. 2&3, ps. 188-207.

Dutton, G.H., 1989; *Modelling locational uncertainty via hierarchical tessellation;* M. Goodchild and S. Gopal, eds., **Accuracy of Spatial Databases**. London: Taylor & Francis, ps. 125-140.

Eisele, C., 1963; *Charles S. Pierce and the problem of map-projection;* **Proc. Amer. Philo. Soc.**, vol. 107. no. 4, ps. 299-307.

Fisher, I., 1942; *A world map on a regular icosahedron by gnomic projection,* **Geographical Review,** October, pp. 605-619.

Fisher, I. and O.M. Miller, 1944; **World maps and globes.** New York: Essential Books.

Fuller, R.B., 1982; **Synergetics: Explorations in the geometry of thinking,** New York: Macmillan, 2 vols.

Goodchild, M. and S. Yang, 1989; *A hierarchical spatial data structure for global geographic information systems.* Santa Barbara, CA: **NCGIA tech. paper 89-5**.

Life, March 1, 1943; article about Buckminster Fuller's *Dymaxion Globe.*

Lucas, D., 1979; *A Multiplication in N-space.* **Proc. Amer. Math. Soc.** Vol 74, no. 1.

DETECTING FEATURES IN LOW RESOLUTION AERIAL IMAGES OF CITY BLOCKS

Bill Sakoda
Joseph G. Hauk

Computer Science Department
State University of NY at Stony Brook
Stony Brook NY 11794
and
Grumman Data Systems
1000 Woodbury Road
D12-237
Woodbury NY 11797

Contact: Bill Sakoda
wjs@sbcs.sunysb.edu
(516) 632-8439

ABSTRACT

This project is a case study of extraction of roads and houses from low-resolution infrared aerial photographs of city block areas. Houses and roads are about 2 pixels wide. Infrared renders houses, connecting driveways, and roads light, with significant blurring. The situation is challenging because of the similar imaging of the objects of interest and the low resolution.

We show how to combine regions from thesholding, a residual-based edgefinder, and spot (house) identification through a modified Gaussian curvature to obtain road networks and houses. A tree growing procedure for aggregating points in the plane is developed, and applied to find smooth trajectories through detected building locations, yielding rows of houses along roads. In addition to proposing a practical method for this problem domain, we hope that this and similar studies contribute to development of techniques for low-level feature extraction and methods for combining them.

INTRODUCTION

This project is a case study of extraction of roads and houses from low-resolution infrared aerial photographs of city block areas. Houses and roads are about 2 pixels wide. Infrared renders houses, connecting driveways, and

Presented at Auto Carto 10, March 23-29 1991, Baltimore MD.

roads light, with significant blurring. The situation is challenging because of the similar imaging of the objects of interest and the low resolution. We are interested in extracting maximum information from low resolution data because this reduces the number of images that need be captured, and because in surveillance applications, low resolution data may be all that are available.

We show how to combine regions from thesholding, a residual-based edgefinder, and spot (house) identification through a modified Gaussian curvature to obtain road networks and houses. A spanning-tree-based procedure for sending smooth paths through points is developed. Such a procedure has numerous applications because significant physical objects tend to have smooth boundaries, while feature detectors often produce fragments. One can deduce the boundaries by reassembling the fragments. This module is tested by applying it to house locations, yielding rows of houses along roads. In addition to proposing a practical method for this problem domain, we hope that this and similar studies contribute to development of techniques for low-level feature extraction and methods for combining them.

Figure 1 summarizes our results.

PREVIOUS WORK

There has been much work in this area, due to the variety and volume of data awaiting availability of practical systems, and suitability of numerous sub-domains as testbeds for different techniques.

Emphasis of work in the area includes low-level primitives (Nevatia and Babu 1980), detection of cultural objects by rectangular or smoothly curving contours (Fua and Hanson 1987), complete systems based on applying special knowledge of particular sub-domains (Huertas et. al. 1987), and general mechanisms for applying knowledge constraints (McKeown et. al. 1985). Most work spans the range from use of low level vision to acquire basic data, to application of domain-specific knowledge, whether it be applied as a special case or through a general mechanism. It is typical for authors to comment on special characteristics making their domain challenging to automatic analysis. The present study considers a particular sub-domain -- low resolution infrared city blocks -- and shows how mathematically well-behaved primitives can be used in conjunction with world constraints to extract features. A general system will function most efficiently with a library of such sub-domains and a kernel of mathematically precise low-level detectors. Binford (1982) argues lucidly that success in large domains using model-based analysis will require strong low-level modules.

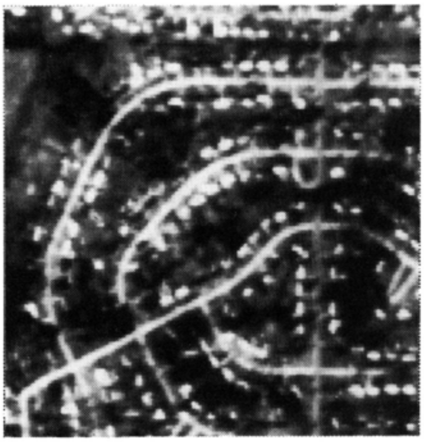


Figure 1a. Source picture CIRCLE. 128 by 128.

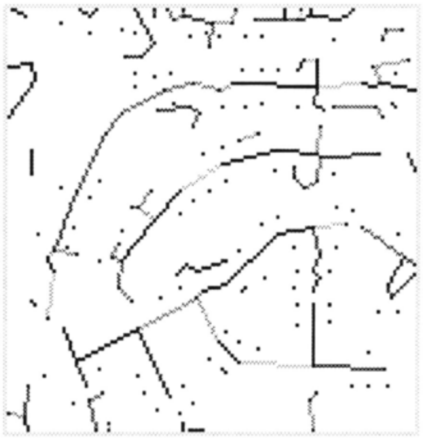


Figure 1b. Features located in CIRCLE. Straight line segments from roads are plotted different shades of gray. Isolated black dots are buildings.

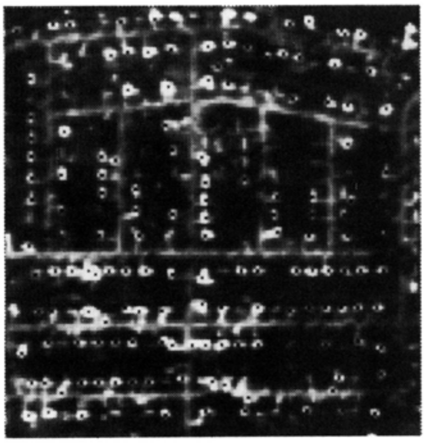


Figure 1c. Picture BLOCK. House Locations found via gaussian curvature marked black. Source image is 128 x 128.

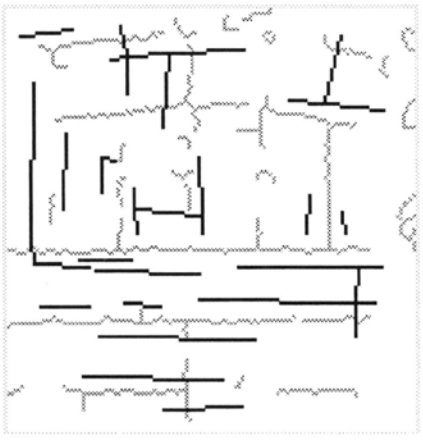


Figure 1d. House grouping results from BLOCK after line fitting (shown in black). Results of road extraction algorithm are shown in grey.

Figure 1: Summary of road and building extraction techniques.

Cultural Feature Detection

Huertas, Cole and Nevatia (1987) demonstrated a system for detection of airport runways from very high resolution photographs. This work showed a nice balance of *simple* but well-considered low (lines) and middle (APAR) level vision, use of hand-tooled high level constraints from the problem domain, and a working demonstration. APARS are approximately parallel edges of opposite contrast (Anti-PARallel), useful for detecting bars or slowly varying ribbon shapes against a contrasting background. They point out that while runways are essentially elongated rectangles, the problem is very challenging because of runway markings, non-uniformity of runway surface (oil spots and shoulders), repair work, vehicles on the tarmac, and intersections. LINEAR (Nevatia and Babu 1980) is used to produce line segments and APARS. A variety of 5 by 5 masks are used to detect edges, which are then thresholded, thinned, linked, and approximated as piecewise linear. APARS are then identified. APAR-based approaches tend to produce many false candidates, especially when a feature has parallel sub-features (eg lines down a runway), and each line can then contributes to many APARS. This is handled by histogramming APAR widths, and selecting candidates with widths appropriate for runways, shoulders and markings. APARS are joined by analyzing continuity, collinearity, and gap texture. Finally, hypotheses of positioning of runway subfeatures are verified from FAA specifications.

Fua and Hanson (1985) used parallel and perpendicular line segments to locate cultural objects in high resolution images. Undersegmentation was resolved by using linking to connect almost-collinear lines, complete corners, and close open-ended U's and parallels. Subsequently (Fua and Hanson 1987), they proposed detecting roads by using linear edge segments to calculate road width and center; fitting a spline to the center; then using the center spline to locate splines for each side of the road. This allows the road to be continued even when one side is lost due to imaging conditions, occlusion, junctions.

Pavlidis and Liow (Pavlidis and Liow 1988) detected regions by following an oversegmented split-and-merge phase with boundary and edge modification based on contrast, boundary smoothness, and image gradient along boundaries.

The integration of top-down and bottom-up analysis has been advocated by many authors. In particular, Matsuyama (1987) presented an image understanding system that generates hypotheses to test for the existence and location objects, according to the results of low-level vision techniques.

Similarly, Nicolin and Gabler (1987) demonstrated a knowledge-based system for interpretation of aerial images of suburban scenes. Their system is divided into several functional units. One unit contains a methods base of low-level image processing techniques and a second unit contains a knowledge base for suburban scenes. The system's control module uses the knowledge base to decide which techniques from the methods base should be applied to the image.

McKeown and Denlinger (1988) constructed a system for high-resolution imagery based on cooperation between a surface correlation tracker and edge tracing. They detect edges using a 5 by 5 Sobel gradient. The correlation tracker, after a design of Quam (1978), looks for patterns such as lane markers and wear patterns. Starting position of the road, its direction, and width are assumed given. The hypothesized road trajectory is tested by pushing a cross-section of the road forward and testing for cross-correlation.

Aviad and Carnine (1988) presented a method for generating hypotheses for fragments of roads, intended to be fed to a road tracker. The Nevatia and Babu (1980) edge finder is used, followed by Road Center Hypothesis detection by antiparallel edges. RCH's are then aggregated by a greedy linker. This is followed by editing by a smoothness checker, and a final linking.

Point Grouping

In our paper, we examine how to group points in the plane (houses locations) into smoothly varying curves that will lend insight into the feature composition of an aerial image. Zahn (1971) applied graph theoretic algorithms to detection of clusters in arbitrary point patterns. By constructing a minimal spanning tree, he is able to cluster dots into groups according to their point density, measured by calculating the local average length of the spanning tree's edges. A histogram of the local point densities is then calculated and categorized. All edges having neighbors of two (or more) different point density categories are deleted. The resulting graph contains a spanning tree for each point cluster.

Stevens (1978) showed that orientation patterns in a field of random dots can be detected by the use of a local support algorithm. Local orientation is found by drawing virtual lines between neighboring points and then searching for the predominant orientation of the virtual lines. For example, Stevens' algorithm can deduce local relationships in a pattern consisting of an original set of random dots together with a duplicated translation, or with a duplicated set expanded about a center. It will also group isolated one-dimensional curves. Since it finds the *major* orientation in two-dimensional neighborhoods, it is not well suited to grouping houses, where there are nearby linear strings of different orientation. Likewise, Zucker (1985) presented an orientation-based process to infer contours from a collection of dots by locally finding the tangent fields.

Tuceryan and Ahuja (1987) performed clustering and linking according to properties of the Voronoi polygons induced by the dots, including area, eccentricity, isotropicity, and elongation. For example, dots around the boundary of a cluster can be identified because they are eccentric within their polygons.

Vistnes (1987) used a statistical model for the detection of dotted lines and curves embedded in a random dot field. His model is based on a local operator that detects regions of differing dot densities.

ROAD AND HOUSE DETECTION

The goal of this phase is efficient generation of a map of as many of the main roads and houses as possible. When in doubt we are conservative, identifying those features in which we have high confidence. We pay particular attention to *connectivity* of the road networks as this is a key semantic feature.

APARS tend to produce disconnected representations at junctions, (e.g., at a Y junction in a road) since the generating parallel edges do not continue all the way to the center of the junction. In contrast thinning naturally preserves the connectivity at the junction. Centers between APARS are much less sensitive to small glitches in data than thinning in finding skeletons for wide objects, but thinning is suitable for the present domain because features are only a few pixels wide. These data are a difficult (though possibly feasible) case for edge linking and APAR detection because very close proximity of houses causes the edge-finder to wander. Our greedy thresholding quickly and simply yields good connectivity over large segments.

Road and house detection runs in 6 steps.

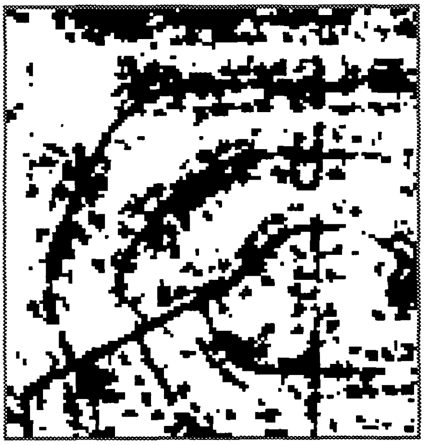
1. (Greedy thresholding): Undersegmented regions consisting of houses, roads driveways, and some adjoining areas are thresholded from the source.
2. (Residual edge cutting): The Lee/Pavlidis/Huang (1988) residual edgefinder is tuned to handle these small-scale data, and edges are used to further segment the regions.
3. (Thinning and small component removal): Resulting regions are thinned. Thinning is careful to respect connectivity, which can now be deduced by local analysis of neighbors. Small components are removed.
4. (Gaussian curvature spot detector): A modified Gaussian curvature spot detector is applied, yielding most house positions.
5. (Trimming): House locations and connectivity of the road network are used to trim the network down to major roads.
6. (Line fitting): The network is decomposed into line segments.

Thresholding

This data set is interesting because neither the regions from thresholding nor edgefinding by themselves yield adequate information about the road networks. A greedy threshold does maintain good connectivity, but blurs the houses into the roads. Adjacent houses blur together, mimicking the linear structure of the roads. More conservative thresholds curtail this aggregation somewhat, but even when the threshold is reduced to the point where the roads begin to disconnect, significant aggregation of distinct features remains. Figure 2 shows the regions obtained at 2 threshold values.

Edgefinding

Our edgefinder is based on the residual technique of Lee, Pavlidis and Huang (1988). They detect edges as zero-crossings of the difference between source image and a regularization of the image. We found that effects of small variations in the image were reduced by applying a mild smoothing to the image first. So, edges are the zero-crossings of the difference of a mildly regularized


(2 a) Image CIRCLE threshold > 80

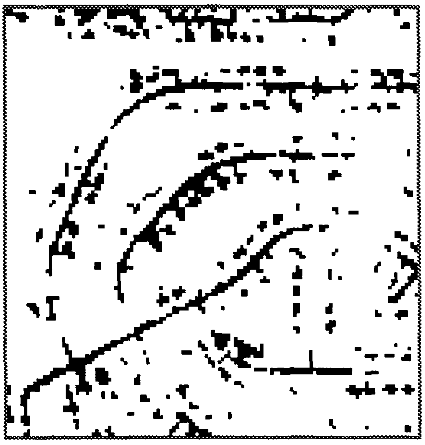

(2 b) CIRCLE threshold > 110

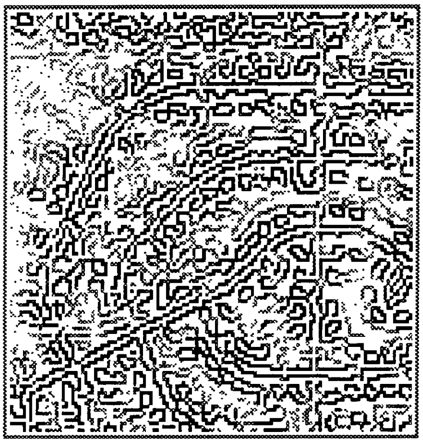
Figure 2: A greedy theshold yield excellent road connectivity but blurs houses into roads. More conservative thresholds yield neater road segments, but leave gaps, eliminate small roads, and still leave some building and driveways attached. We take the undersegmented greedy image and use edges to refine it.

image ($\beta = 5.0$), and a smoother image ($\beta = 1.0$). Zero-crossings are thresholded by slope.

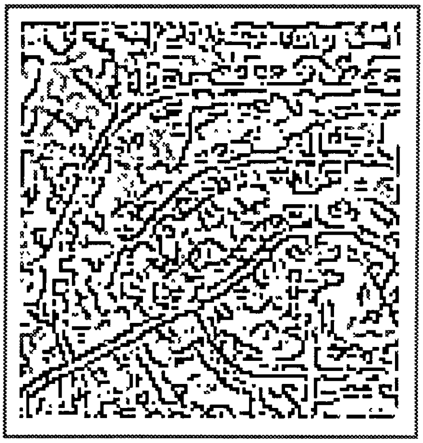
We compared Canny and residual edgefinders at different resolutions. The residual finder tended to generate more closed or almost-closed contours around small features like houses, which is especially useful in the trimming technique we are using. Figure 3 shows Canny and residual edges.

A common method for recording an edge map is to mark edge pixels on a raster the same size as the source image. Here one might choose to mark the edge on the darker side, the lighter side, or on the side nearer zero. Doing so in this case blurs nearby linear features (eg adjacent roads) together. Fortunately zero-crossings have more structure than this - they form contours. A zero-crossing falls between raster positions having positive and negative residual values. We use a zero-crossing tracker which walks this boundary, breaking zero-crossing contours into smooth segments. A raster twice the size of the source is used, with cells having even coordinates holding the source picture, and with zero-crossing information stored between.

This data structure is now used to prune the regions. Since we are trying to preserve the lighter structures, the zero crossing segment walker sets every pixel on the darker side of a zero-crossing to black. This corresponds to a cut when lighter areas are chosen by thresholding. Figure 4 shows the result of removing edge points.

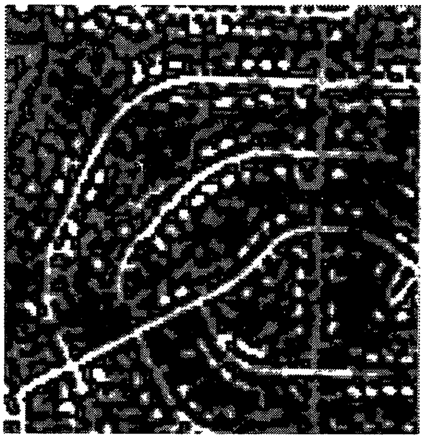


(3 a) Edges from residual edge finder for image CIRCLE

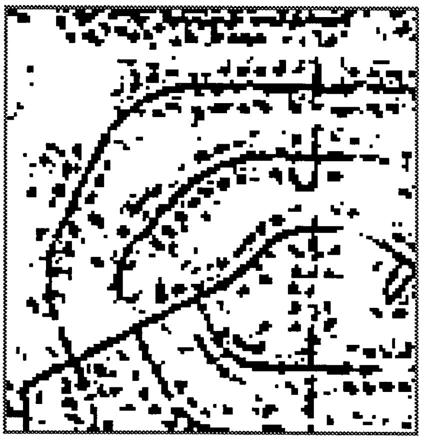


(3 b) Canny edges, $\sigma = 1.0$

Figure 3: Residual and Canny edges. Stronger edges are plotted darker. We found that the residual edges yielded more road boundaries, and tended to trace closed contours around houses.



(4 a) Image CIRCLE with edge points removed (black)



(4 b) Image CIRCLE: edge points removed from greedy threshold

Figure 4: Thresholded regions have good connectivity; Edges are sparse but have good spatial accuracy.

Residual edges in this example accurately delineate many physically significant boundaries but are sparse and would present a difficult case for a purely edge-based technique. A purely edge-based analysis *might* be possible and would be very interesting.

Thinning

The image is thinned (Pavlidis 1982), and small components are removed. The results are in Figure 5.

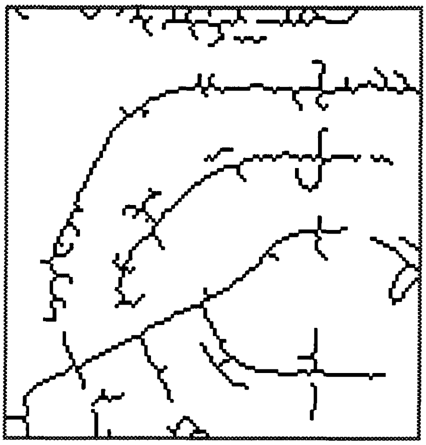


Figure 5: Image CIRCLE after thinning and removal of small components

Gaussian curvature spot finder

Gaussian curvature can be used to construct an efficient operator which responds strongly to small bright spots (eg, houses around 2-3 pixels wide, as they are in our source image), but does not respond to straight edge data (roads).

Gaussian curvature of the gray level picture g(x,y) is given by:

$$\frac{\frac{\partial g^2}{\partial x^2}\,\frac{\partial g^2}{\partial y^2} - \left(\frac{\partial^2 g}{\partial x \partial y}\right)^2}{\left(1 + \left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2\right)^2}$$

(See, e.g., Spivak (1970) or Horn (1986). We have been using the numerator of this expression to locate spots. It can be rewritten:

$$\left[\frac{\partial}{\partial x}\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)\right] \times \left[\frac{\partial}{\partial y}\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)\right]$$

In this rendering as cross-product of directional derivatives of gradient vectors we can see directly why there is no response to straight edges. All gradients point normal to the direction of the edge. Therefore derivatives also point in this direction, yielding a zero cross-product. This quantity is invariant of orientation of the coordinate system because the Gaussian curvature and denominator both are invariant.

In discrete image space at point (x, y) this curvature c can be computed as:

$$dxy = \left[\frac{1}{4}\right] \left(\left[g_{x+1,\,y+1} - g_{x+1,\,y-1}\right] - \left[g_{x-1,\,y+1} - g_{x-1,\,y-1}\right] \right)$$
$$d\_sq\_x = g_{x+1,\,y} + g_{x-1,\,y} - 2\,g_{x,\,y}$$
$$d\_sq\_y = g_{x,\,y+1} + g_{x,\,y-1} - 2\,g_{x,\,y}$$
$$c = d\_sq\_x \cdot d\_sq\_y - dxy^2$$

Our Gaussian curvature module finds bright spots by first convolving the input picture with a gaussian; then finding points of high curvature. A spot is reported at locations whose curvature is greater than a specified threshold, and not less than the curvature of its 4 compass neighbors. This tends to mark a single pixel for each bright spot. Figure 1c shows the results of the Gaussian curvature operator.

Trimming via Connectivity analysis

At this point the constructed network has much of the connectivity structure of the underlying roads. Junctions can be located by locally counting neighbors. One must allow for diagonal connectivity to a neighbor if there is no 4-connectivity, as in Figure 6(a). Junctions of degree greater than 3 may be spread over neighboring pixels (Figure 6b).

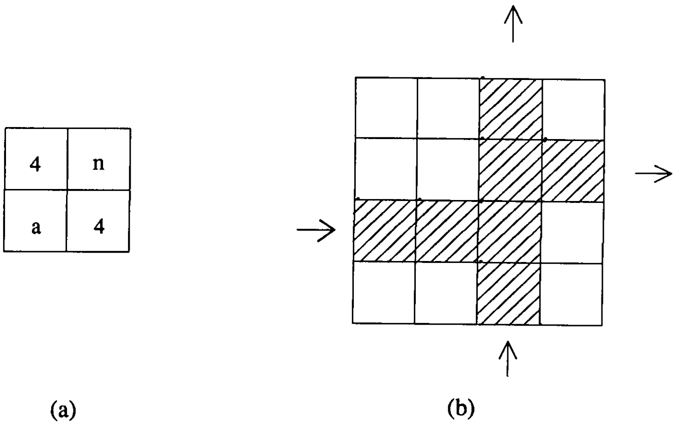
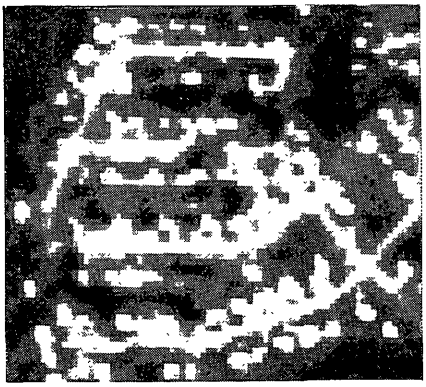
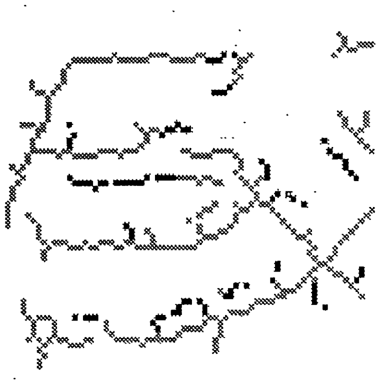
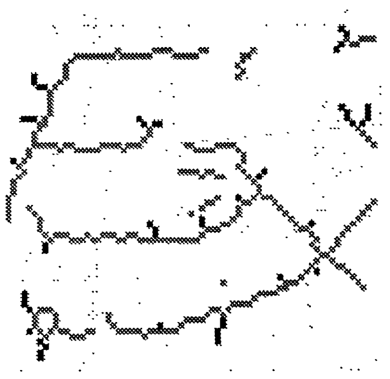


Figure 6: (a): In looking for junctions, cell "a" should count "n" as a neighbor only if neither of their common 4-neighbors is occupied. (b): When ≥ 4 roads meet, the junction can be spread over nearby cells.

Strings of nearby houses tend to blur together, producing linear bright strips which mimic road structure. A two-stage preening is now undertaken to eliminate these. First, starting from each house point, road pixels are deleted back up to a distance 7, but not past junctions. Stopping at junctions deletes driveways up to a main road without interrupting it. Then short stubs are deleted, by starting at endpoints and looking for a junction within 5 pixels. If a junction is found, the segment from the endpoint the junction is deleted. Figure 7 shows the results.


(7 a) Source picture ANGLE; 70 by 80. May be better viewed from a distance (like Harmon's Abraham Lincoln).


(7 b) Spots from Gaussian curvature detector in white. Linear segments attached to houses, to be deleted, in black.


(7 c) Remaining short stubs to be deleted.

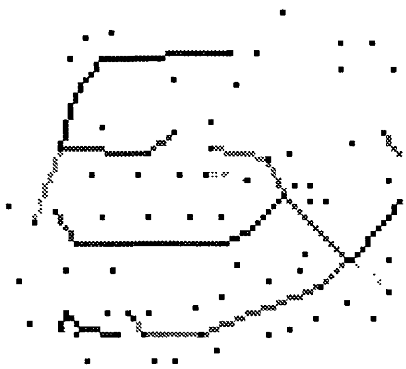

(7 d) Final results after line fitting.

Figure 7: Deletion of driveways and aggregated houses. Starting from house locations, road pixels are traced and deleted up to distance 7, but not past junctions. (b): Result after thresholding, edge cutting, thinning, small component deletion, with road segments to be deleted in black. (c): Remaining short stubs to be deleted, in black.

Line Fitting

Straight lines are fit to the thinned segments by testing line segments between successively more distant data points, and breaking at points of sufficiently large maximum distance between the test line and data (Pavlidis 1988b).

DOT GROUPING

In this section, we develop an algorithm for grouping points in two-dimensional Euclidean space, and apply it to detecting the curvilinear structure of houses lying beside roads. The algorithm takes as input a set V of points from R^2 (the detected locations of houses in our application), and constructs a forest joining certain vertices in V. The forest is grown as a sequence of trees, and each tree is grown by adding edges. The cost metric $C(T,v,w)$ designates the cost, possibly ∞, of adding edge (v,w) to the partially constructed tree T. The algorithm is structurally similar to Prim's (1957) greedy algorithm for constructing a minimum cost spanning tree. Our model differs in allowing the metric to be a function of the partially constructed tree. A great deal of control over the grouping can be exercised by varying the metric.

Grouping algorithm

Let $T_{i,j}$ denote the i-th tree after j edges have been added.

1) (Start a new tree): Let i be the number of trees constructed so far. We let $T_{i+1,1}$ consist of the edge joining a pair of vertices at minimum Euclidean distance, among all vertices not contained in any tree. If no edge is found, the construction is complete.

2) (Grow the current tree): Among all vertices w not in any current tree, and all vertices v in the current tree T_i, find a pair (v_0,w_0) minimizing $C(T_i,v_0,w_0)$. Add it to T_i and iterate this step. If no finite-cost addition can be found, go to (1) to start the next tree.

We have found it computationally and semantically advantageous to disallow edges longer than a parameter $D_{\max}$. This can be subsumed in the model by assigning any edge of length greater than $D_{\max}$ infinite cost, and leads to an implementation where vertices can be assigned to local buckets of size $2D_{\max}$ by $2D_{\max}$, and search for an appropriate neighbor of vertex v can be constrained to at most 4 buckets.

We now demonstrate how our state-dependent metric can be used to advantage. Let

$$C_{curve,\alpha}(T,v,w) = \alpha * Dist(v,w) + (1-\alpha) * angle(T,v,w) .$$

$angle(T,v,w)$ is the absolute value of the angle (in degrees) that is formed between the new branch and the neighboring branch (already included in the current tree T) having the most similar orientation. Parameter α determines the relative weighting between orientation and proximity, with $\alpha = 1$ being the simple minimum Euclidean distance metric. The value $\alpha = .9$ has produced good results in grouping houses in this dataset.

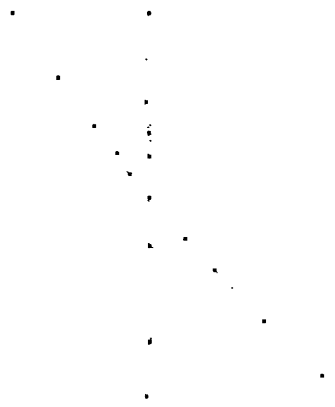


Figure 8a. Example 1: Minimum Euclidean distance spanning tree.

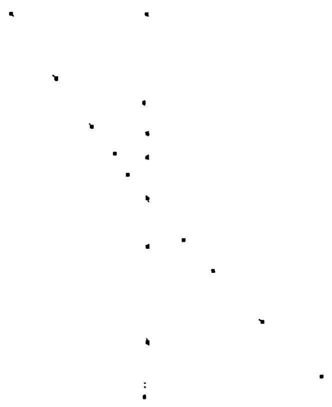


Figure 8b. Example 1: Tree constructed by orientation-sensitive metric $C_{curve,\alpha=.9}$

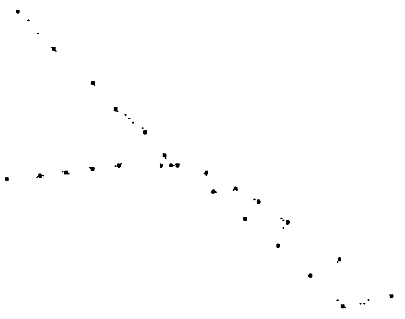


Figure 8c. Example 2: Minimum Euclidean distance spanning tree.

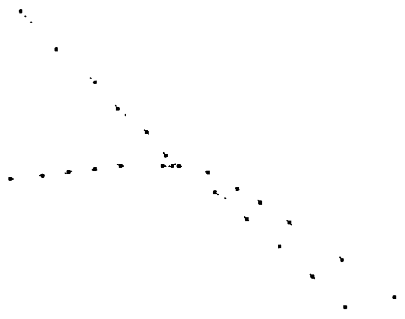


Figure 8d. Example 2: Tree constructed by orientation-sensitive metric $C_{curve,\alpha=.9}$

Figure 8: Two examples of smooth curve tracking ability of the orientation sensitive metric. Left column shows the minimum cost spanning tree under the Euclidean metric; right column shows the metric $C_{curve,\alpha=.9}$ combining distance and change in orientation.

Figure 8 shows how use of orientation information can assist in tracking smooth intersecting curves: $C_{curve,\alpha=.9}$ follows the curves (Figure 8 b, d) better than the pure proximity-based metric (Figure 8 a, c).

Figure 9a shows the result of applying $C_{curve,\alpha=.9}$ to detected building locations in our dataset. Much of the linear structure of the house groups is captured, but some of the branches selected do not lie parallel to the nearby road but instead cross perpendicular to it. This occurs when houses on opposite sides of the same street are close enough to form a link. These stubs can be avoided by looking ahead for a smooth extrapolation: if we are trying to extend from vertex v in the current tree to a new vertex w, the extension is allowed only if there is an additional vertex x with the angle between (v,w) and (w,x) close to 180 degrees. This yields a new cost metric:

$$C_{lookahead,\alpha,\varepsilon}(T,v,w) = \begin{cases} C_{curve,\alpha}(T,v,w) & \textit{if there exists a vertex } x\textit{, with the angle between } (v,w) \textit{ and } (w,x) \textit{ differing from 180 degrees by at most } \varepsilon; \\ \infty & \textit{otherwise.} \end{cases}$$

This metric prevents the growth of branches that form stubs, but keeps corners and crossings intact as shown in Figure 9.

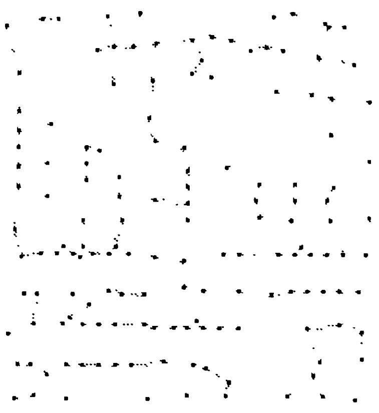


Figure 9a. $C_{curve,\alpha=.9}$ tracks curves well, but tends to generate short cross links between segments.

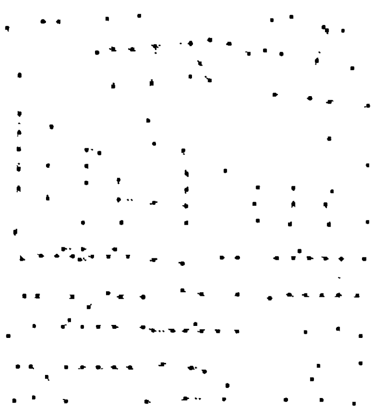


Figure 9b. $C_{lookahead,\alpha=.9}$ inhibits cross links by requiring that there be at least 2 adjacent edges lying nearly along a straight line.

Finally, we calculate the two most frequently occurring house link orientations and delete any links whose orientation differs significantly. This is reasonable since communities generally have roads that run in two primary directions. Then a simple line fitting program (Pavlidis 1988) is used to straighten the house clusters. The house clustering results together with the results of the road extraction algorithm are shown in Figure 1.

Future work on clustering

The next step would be to integrate the roadfinding with building grouping. Our building groups occasionally cross roads, and this could be inhibited. Where houses are relatively sparse the roadfinder alone tends to work well. With dense houses, the similarity of house and road imaging tends to complicate the roadfinding, but the houses group well, providing additional semantic clues. Grouping algorithms tend to be more tolerant to noise in the case of high density. The algorithms should synergize well, as in most places they agree, but there are places in the image where one algorithm has strongly located a road (or road segment) in which the other algorithm has difficulty or misses completely.

SUMMARY

The residual edge-finder tends to produce strong contours around small objects. For this data set, these edges can be used to significantly improve segmentation of low resolution road networks obtained from thresholding. A variant of Gaussian curvature is effective in locating buildings. We have developed a spanning tree technique sensitive to angles between branches, and shown it to be effective in detecting smoothly varying trajectories through given points.

ACKNOWLDEGEMENTS

We are grateful to Dr. Herb Tesser for suggesting this problem, and to Theo Pavlidis for many useful discussions and suggestions.

REFERENCES

Aviad, A. and Carnine, P. D. Jr (June 1988): "Road Finding for Road-Network Extraction," *Proceedings: IEEE CVPR*, pp. 814-819.

Binford, T. O. (1982): "Survey of Model-Based Image Analysis Systems," *Int. J. Robotics Res.*, vol. 1, no. 1.

Fua, P. and Hanson, A. J. (December 1985): "Locating Cultural Regions in Aerial Imagery Using Geometric Clues," *Proceedings: Image Understanding Workshop*, pp. 271-278.

Fua, P. and Hanson, A. J. (1987): "Using Generic Geometric Models for Intelligent Shape Extraction," *Proceedings: Image Understanding Workshop*, pp. 227-233.

Horn, B. K. P. (1986): *Robot Vision*, MIT Press.

Huertas, A., Cole, W., and Nevatia, R. (July 1987): "Detecting Runways in Aerial Images," *Proceedings: AAAI-87*, pp. 712-717.

Lee, David, Pavlidis, T., and Huang, K. (1988): "Edge detection through Residual Analysis," *Proceedings: IEEE CVPR*, pp. 215-222.

Matsuyama, T. (1987): "Knowledge-Based Aerial Image Understanding Systems and Expert Systems for Image Processing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, pp. 305-316.

McKeown, D., Harvey, W. A., and McDermott, J. (1985): "Rule-Based interpretation of Aerial Imagery," *IEEE Trans. PAMI*, vol. PAMI-7, pp. 570-585.

McKeown, David M. and Denlinger, Jerry L. (June 1988): "Cooperative Methods for Road Tracking in Aerial Imagery," *Proceedings: IEEE CVPR*, pp. 662-672.

Nevatia, R. and Babu, R. (1980): "Linear Feature Extraction and Description," *CVGIP*, vol. 13, pp. 257-269.

Nicolin, B. and Gabler, R. (1987): "A Knowledge-Based System for the Analysis of Aerial Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, pp. 317-329.

Pavlidis, Theo (1982): "An Asynchronous Thinning Algorithm," *CGIP*, vol. 20, pp. 133-157.

Pavlidis, Theo (1988b): Personal communication.

Pavlidis, Theo and Liow, Yuh-Tay (June 1988): "Integrating Region Growing and Edge Detection," *Proceedings: IEEE CVPR*, pp. 208-214.

Prim, R. C. (1957): "Shortest Connecting Networks and some Generalizations," *BSTJ*, vol. 36, pp. 1389-1401.

Quam, Lynn H. (May 1978): "Road Tracking and Anomaly Detection in Aerial Imagery," *Proceedings: Image Understanding Workshop*, pp. 87-100.

Spivak, M. (1970): *Differential Geometry*, vol. 2, p. 95.

Stevens, K. A. (1978): ''Computation of Locally Parallel Structure.,'' *Biol. Cybernetics*, vol. 29, pp. 19-28.

Tuceryan, Mihran and Ahuja, Narendra (1987): ''Extracting Perceptual Structure in Dot Patterns: An Integrated Approach,'' *University of Illonois at Urbana-Champaign Technical Report*, vol. UILU-ENG-87-2206.

Vistnes, Richard (1987): ''Detecting Dotted Lines and Curves in Random-Dot Patterns,'' *Image Understanding*, pp. 849-861.

Zahn, Charles T. (1971): ''Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters,'' *IEEE Trans. Computers*, vol. C-20, pp. 68-86.

Zucker, Steven W. (1985): ''Early Orientation Selection: Tangent Fields and the Dimensionality of Their Support,'' *McGill University Technical Report*, vol. TR-85-13-R.

Automatic Digitization of Large Scale Maps

Andreas Illert

Institute of Cartography
University of Hannover
Appelstrasse 9A
3000 Hannover 1
Germany

Abstract

This paper describes a software system for automatic digitization of large scale maps. The system is capable of converting raster data into structured vector data. Strategy and configuration of the software are explained. Some tests on German maps are introduced in brief.

1. Introduction

A GIS is a computerized database management system used for the capture, storage, retrieval, analysis and display of spatial data. Of these items, especially storage and analysis of mass data are what makes a GIS such a powerful tool. However, the problem remains to get the mass data into the computer.

Digital data flowing from the environment into the computer may be managed solely by remote sensing techniques. Satellite images supply up-to-date information, but such information is restricted by pixel resolution, multispectral classification and visibility of the features. Terrestrial topography produces data which are very accurate, but field work is time-consuming and expensive. For these reasons, digitization of areal photography or existing maps has become the most common method for data capture in large scale mapping.

As mentioned above, data capture is defined as part of a GIS. If you look around at the tools GIS products offer, most of them prefer manual digitization by hand-held cursor. Such conventional systems are easy to handle and adapt well to different tasks. Unfortunately, manual digitization is a laborious procedure, and human labour is costly today.

To overcome this problem some companies have provided semi-automatic systems which support manual digitization by line following algorithms. The operator has just to indicate the beginning of a line, the computer then traces the line until the next node. Line following systems are quite effective with isoline maps, but interaction increases with the number of nodes on complex maps.

Data capture by scanner is very fast and requires a minimum of human interaction. Result of scanning is a raster image. Pixel format performs excellent with two dimensional coverages in small scales, but whenever linear features, topology or non-geometric attributes have to be handled vector data are better suited. Especially in large scale mapping, vector format satisfies the demands much better than raster data. Thus, scanned data needs to be converted into structured vector data.

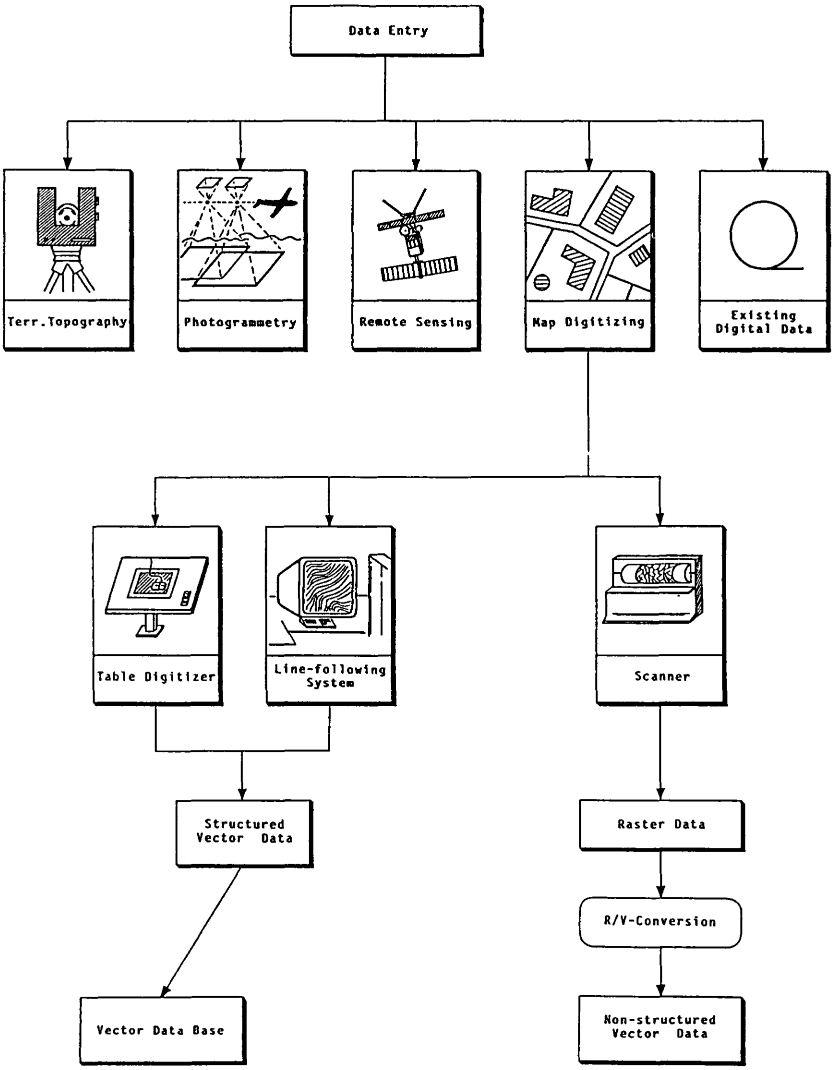


Figure 1 Data Entry to GIS */Lichtner,Illert 1989/*

At present, visual recognition of printed texts with computers seems to have no problem any more. On the other hand, automatic interpretation of drawings is still a matter of research. Nevertheless, some algorithms of text recognition may as well be applied on maps. Scientists at the Institute of Cartography, University of Hannover have developed a software system which combines common methods of Optical Character Recognition with specific algorithms for mapping applications. Concepts and results will be described in the following.

2. Strategy

Input of the recognition system is a raster image, while output is structured vector data. In other words the system has to handle both raster- and vector data, including raster-to-vector-conversion. Pattern recognition methods may be performed using both types of data.

Automatic interpretation of maps splits into two parts: First the contents of a raster image have to be broken down into graphic primitives, such as arcs, letters and symbols. Prior to application of character recognition procedures, the texts and symbols are separated from the rest of data. Next the shape of the features has to be described by numerical characteristics. Finally the features are classified according to these characteristics using methods of statistical pattern recognition. The features are treated one by one without considering any context.

In the second step the recognition system combines primitive elements into objects of a higher level. In contrast to the first step, not isolated elements but relations between features are examined. Classification of relations and structures is performed using rules which build a model of the map. Two approaches are known: the knowledge-based approach and the procedural approach.

The knowledge-based approach takes advantage of Artificial Intelligence tools. A knowledge-based system consists of a knowledge base and an inference engine. The knowledge base holds rules and facts about map features. The inference engine enables classification by matching the rules with the data. Sequence of the rules and facts in the knowledge base should play no role at all. Therefore updating of the system is quite easy. The knowledge-based system does not require information about a certain solution strategy — the inference engine tries to reach a goal without human help so long as the knowledge base supplies sufficient information. Thus knowledge-based systems are very flexible and user-friendly.

Unfortunately, the inference engine slows down rapidly with the increasing number of rules. Efficient applications are limited with nowadays technologies to a number of about 300 rules. If you consider the variety of graphic representations in map design, this is much less than required. Therefore the performance of the inference engine has to be improved by information which defines the combination of rules. As a result, application of rules tends towards a fixed sequence, and the knowledge-based system converts into a procedural system.

A procedural system follows the principles of traditional software engineering. The programmer evaluates a strategy by arranging rules in a fixed sequence, which turns to be an algorithm. Algorithms run very fast and effectively — depending on the skills and experiences of the programmer. Building up a procedural system for automatic digitization of maps means to develop specific algorithms for different representations of spatial features. The expenditure on software development is very high. Once a procedure is established its application is limited to a specific type of map.

Recently procedural systems seem more suitable to practical applications with large sets of data than knowledge-based systems. In this context, the system developed at Hannover is oriented to the procedural approach. Nevertheless, research on knowledge-based methods continues and may replace the traditional methods in future */Meng,1990/*.

3. The recognition procedure

At the Institute of Cartography, University of Hannover a software system named CAROL (Computer-assisted Recognition of linear features) has been developed during the last five years. Goal of the system is automatic digitization of large scale German maps. As mentioned above, the system follows a procedural strategy.

3.1 Data aquisition

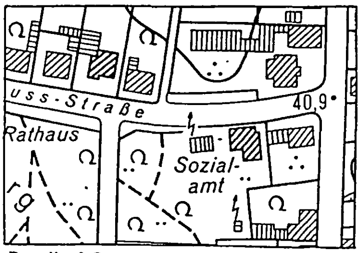
First of all, the map has to be scanned. Scan resolution relies on the type and quality of map. With low resolution, details might get lost. With high resolution, the amount of data and noise within the raster image will increase. In most applications a resolution of 50 μm (500 dpi) proved sufficient. The maps dealt with in Hannover used to be black and white, so raster data is organized in binary format. In case of colour maps, binary images are obtained either by scanner firmware (colour separation) or image processing (multi-spectral classification).

3.2 Raster-to-vector conversion

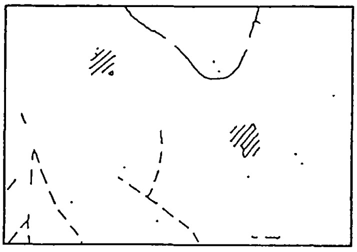
Raster-to-vector conversion is performed by the software tool RAVEL */Lichtner,1987/*. Using distance transformation and topologic skeletonization algorithms, the vectorization program extracts lines from the raster image and arranges them in an arc-and-node-structure. At further steps, connected lines are linked to line networks. Iconic polygons are computed from the line network. So far, only geometry and topology of the map are known. The map graphics are broken down into primitive elements.

3.3 Segmentation

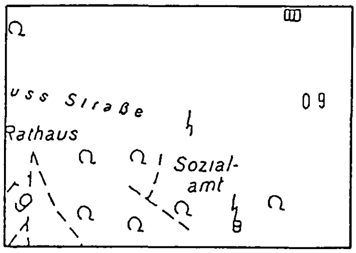
To enable character recognition, letters and symbols have to be composed from arcs and separated from the rest of line graphics. Since such features are in most cases isolated networks, segmentation can be carried out quite easily by checking the size of a circumscribing rectangle. If texts and symbols intersect with the line network, additional information like line width or straightness have to be considered. Characteristic of segmentation procedures is that they take advantage of simple classification operations. Figure 2 demonstrates the effect of some threshold operations on the vector data. Segmentation procedures structure the data in a rough way and therefore help to reduce the expenditure on detailed classification.



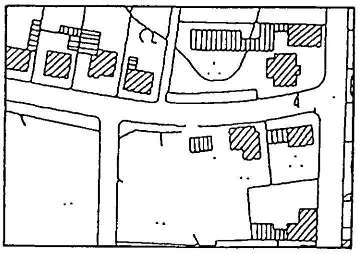
a) Detail of German Base Map scale 1 : 5000



b) Line Width ≥ 6 Pixels



c) Extension of Line Networks in X,Y ≤ 40 Pixels



d) Complementary to c)

Figure 2 Segmentation of unstructured vector data

3.4 Recognition of texts and symbols

Next isolated texts, symbols and numbers have to be classified. This task is similar to the objectives of Optical Character Recognition, and so Cartography may make use of existing methods. Template matching counts as the easiest approach. A template is defined as an ideal pattern of the class. This template is matched against the raster image. The procedure reveals high success rates, although computations are very simple. Unfortunately, template matching works only on features with uniform size and rotation. Thus the method does not satisfy the demands of many applications.

So procedures have to be considered which extract characteristics independent of size and rotation. A method adopted in the CAROL system is expansion of the contour in a Fourier series */Zahn,Roskies 1972/*, */Illert,1988/*. While tracing the contour of a feature the angular change is summed up and perceived as a function of arc length from the starting point. This angle versus length function is expanded in a Fourier series. The Fourier descriptors (i.e. amplitudes and phase angles) are taken as chracteristics. Figure 3 demonstrates the method by reconstructing the original contour polygon from Fourier Descriptors of increasing degree. An expansion up to degree ten already yields sufficient information for shape recognition. In addition to Fourier Descriptors further characteristics like number of nodes or arcs may be included to improve the results of classification.

The classification itself is based on statistical analysis. The n characteristics of a feature define its location in the so-called n-dimensional feature space. Similar fea-

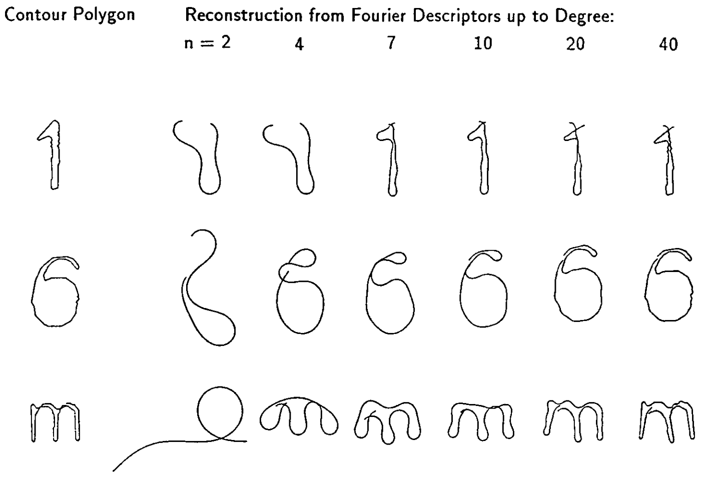


Figure 3 Expansion of contour polygons in a Fourier series

tures of a common class produce clusters in feature space. Therefore a class may be described by parameters of normal distribution. Limitations to the statistical model lead to different classification methods, such as maximum-likelihood-classification or minimum-distance-classification. However, experience reveals that success of classification depends much more on the choice of suitable characteristics than on the statistical model.

3.5 Analysis of complex features

The preceding steps structured the data into arcs, letters and symbols. Now these primitive features have to be combined into objects of higher level. The structure of a map feature in regard to its primitive components is reflected in the map legend. Some graphic structures are common to a lot of map types (e.g. dashed lines, hatching etc.), but a good part is unique for a special type of map. In the CAROL system, procedural analysis of complex features is performed by algorithms like:

— recognition of dashed lines : The system examines the data for repeated sequences of dashes, dots and gaps.

— recognition of hatched areas : The system extracts groups of parallel lines and computes an outline polygon.

— combine symbols in strings (digits to numbers, letters to texts) : The system checks relative position, rotation and size. Feature codes may be changed due to context (e.g. Number '0' and uppercase 'O' or number '1', uppercase 'I' and lowercase 'l').

— Decoding of attributes: Texts, numbers and symbols are assigned to spatial features (lines, polygons, points)

The set up of the procedure, the choice of parameters and the sequence of algorithms should be supervised by an expert. Extensive installation work is necessary whenever the procedural system faces a new type of map.

To enable knowledge-based interpretation, basic rules have to be derived from the algorithms and put into a knowledge base. When applied to a set of arcs, nodes and texts, the inference engine performs the interpretation task. With a global knowledge base no specific set-up would be required, but on the other hand technology does not yet support such ideas.

4. Applications

4.1 Hannover town plan scale 1 : 20.000

The city of Hannover has published a town plan at the scale of 1 : 20.000. Size of the map is 120 x 90 cm^2. The map is printed in 12 colours. Recently local authorities have introduced vector-based GIS software. One of its applications will be thematic mapping. In this context the town map acts as topographic base map.

The City of Hannover, Department of Cartography maintains about 20 printing separates, each of them showing a certain category of features like public buildings, industrial plants, forests or hydrography. Ten of these black-and-white separates were scanned with a resolution of 50 μm, resulting in ten raster images of 25.000 x 18.000 pixels. Raster-to-vector-conversion yields ten sets of vector data, either centre lines (in case of linear features like isolines or small rivers) or contour polygons (in case of areal features like buildings, forests etc.). The centre lines are organized in an arc-and-node structure, whereas polygons are structured hierarchically in regard to feature outlines and enclosed blank areas. Finally, lines are smoothened, and the data sets are merged by affine transformation. The whole process took about one week on microcomputer equipment, resulting in a data base of about 300.000 lines and 50.000 polygons.

4.2 Isoline maps scale 1 : 5000

German isoline maps at the scale 1 : 5000 show topography complementary to the ground situation in base map 1 : 5000. The graphic of isoline maps comprises solid lines with height numbers (height interval 10 m), dashed intermediate lines (intervals 5 m, 2,5 m or 1 m depending on gradient), height spots with numbers and slope symbols.

Computation of a DTM requires height data in digital form. For that the map sheets were scanned and vectorized. Next dashed lines were recognized. Then segmentation operations by parameters *number of nodes, extension in X* and *extension in Y* help to subdivide the data in slope symbols, isolines and numbers. Recognition of digits zero to nine is performed using Fourier Descriptors. After classification the digits were linked to height numbers and assigned to the 10 m isolines or to height spots respectively. Finally, height values have to be assigned to the intermediate isolines through interpolation within the 10 m intervals.

Interactive work is reduced to a maximum of one hour for each map sheet of 40 x 40 cm^2. The procedure is detailed in /*Yang,1990a*/.

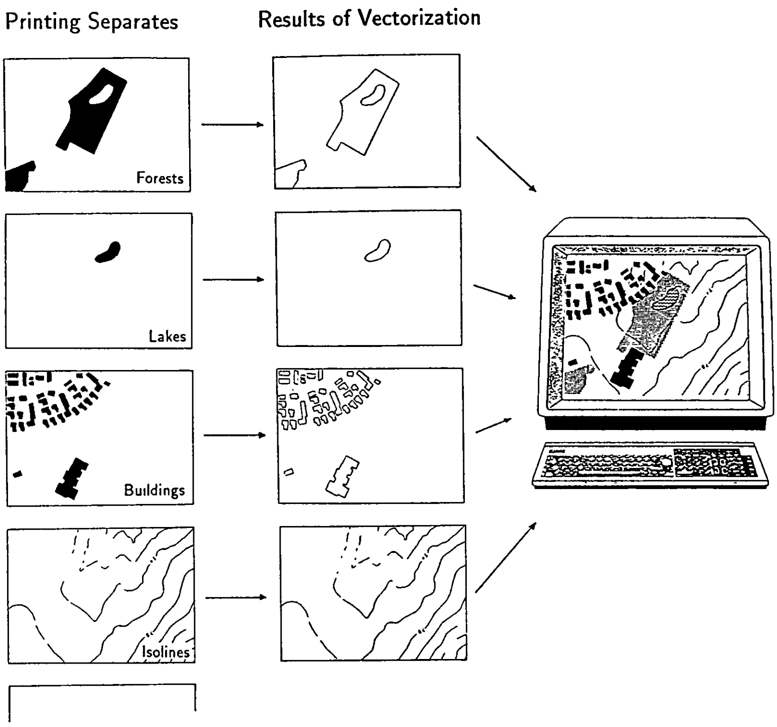


Figure 4 Digitization of Hannover town plan scale 1 : 20.000

4.3 German base map scale 1 : 5000

The base map covers the whole area of West Germany with few exceptions in the south. Ground situation is displayed in detail. The features like buildings, roads or forests are hardly affected by generalization due to the large scale. By this the map is an ideal source for GIS data bases.

The recognition procedure is set up as explained in section 3. Scanning and vectorization of a 40 x 40 cm^2 map sheet produce a set of unstructured vector data, ranging from 20.000 arcs in rural areas to 100.000 arcs in densely populated areas. Recognition of texts and symbols has to classify about 80 different features. Algorithms have been established to analyse some of the most common map features, such as

buildings	(hatched polygons)
forests	(polygons + texture of tree symbols)
meadows	(polygons + symbols: two neighbored dots in level)
gardens	(polygons + symbols: three dots arranged in a triangle)
roads	(long and narrow polygons, inside blank or street name)

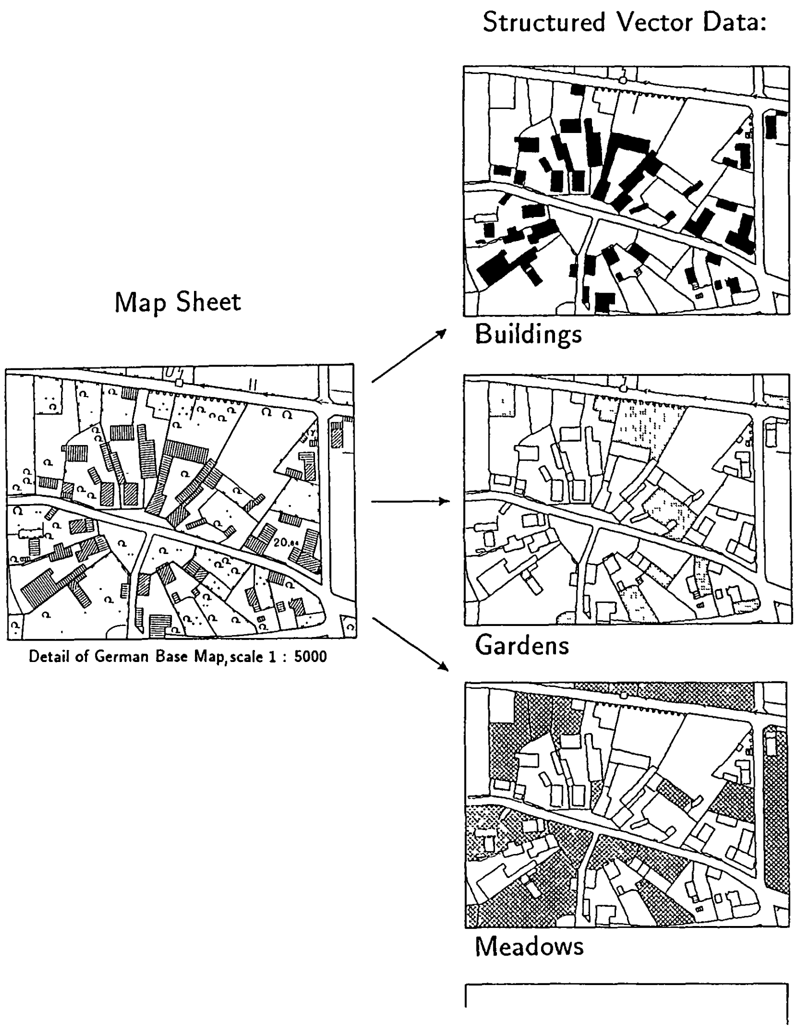


Figure 5 Digitization of German base map scale 1 : 5000

Examples are displayed in Figure 2 and 5.

First tests have been carried out with data aquisition for the ATKIS system of German Surveying Agencies which requires information of some 10.000 map sheets. Automatic digitization reveals success rates of 80 to 95 % (see Figure 5) */Illert,1990/*. However some problems may still arise:

1. Geometry of vector data obtained by scanning has to be enhanced to meet the high quality standards of German cadastre.
2. The classification software should be embedded in a CAD system to support interactive editing of errors during the recognition process.
3. Success rates rise with complexity of algorithms, but on the other hand the system becomes less flexible in regard to application on different map types. Because of that the knowledge-based approach should be kept in mind.

5. References

Grünreich,D. (1990) Das Projekt ATKIS: Konzeption und erste Erfahrungen aus der Aufbauphase des digitalen Landschaftsmodells 1:25000 (DLM25). Proceedings XIX FIG Congress, Helsinki/Finland June 1990, Commission 5, pp. 152-163

Illert,A. (1988) Automatic Recognition of Texts and Symbols in Scanned Maps. Proceedings EUROCARTO SEVEN, Enschede, The Netherlands Sept.88, ITC Publication No.8, pp.32-41

Illert,A. (1990) Automatische Erfassung von Kartenschrift, Symbolen und Grundrißobjekten aus der Deutschen Grundkarte 1:5000 Wissenschaftliche Arbeiten der Fachrichtung Vermessungswesen der Universität Hannover, Nr.166, 1990

Lichtner,W. (1987) RAVEL — Complex software for Raster-to-vector-conversion. Proceedings EUROCARTO VI, Brno/ Czechoslovakia 1987

Lichtner, Illert (1989) Entwicklungen zur kartographischen Mustererkennung. in: Geo-Informationsystems, Applications — New Trends. Edited by Schilcher/Fritsch, Wichmann-Verlag, Karlsruhe 1989, pp. 283-291

Meng,L. (1990) Potentialities of Quintus PROLOG in Cartographic Pattern Recognition. Proceedings EUROCARTO VIII, Palma de Mallorca / Spain, April 1990

Yang,J. (1990a) Automatische Erfassung von Höhenlinien mit Verfahren der Mustererkennung. Nachrichten aus dem Karten- und Vermessungswesen, Series I, No. 105, Frankfurt am Main 1990

Yang,J. (1990b) Automatic data capture for polygon maps from scanned data. Proceedings XIX FIG Congress, Helsinki/Finland June 1990, Commission 5, pp. 522-531

Zahn,Roskies (1972) Fourier Descriptors for Plane Closed Curves. IEEE Transactions on Computers, Vol C-21, No.3, March 1972 ,pp. 269-281

THEMATIC MAPPING FROM IMAGERY: AN ASPECT OF AUTOMATED MAP GENERALIZATION

Minhua Wang[1], Peng Gong[2] and Philip J. Howarth[1]
Earth-Observations Laboratory,
Institute for Space and Terrestrial Science
at
[1]Department of Geography, University of Waterloo
Waterloo, Ontario, Canada N2L 3G1 and
[2]4850 Keele Street, North York, Ontario, Canada M3J 3K1

ABSTRACT

Map generalization, as a means of portraying the complex real world, has to date been confined to the manipulation of map data. With the advent of new data acquisition techniques, particularly remote sensing, data sources for spatial analysis have greatly increased. However, map generalization from image data is a challenging problem. In this paper, the conceptual and technical problems in generalizing cartographic objects from remote sensing imagery are addressed. A two-stage generalization framework is proposed for thematic mapping from imagery. Specific interest is focused on mapping land use from SPOT satellite imagery.

INTRODUCTION

Thematic mapping is a major activity in both cartography and remote sensing. However, due to independent developments in remote sensing and cartography, the theoretical bases for thematic mapping are considerably different. In cartography, thematic mapping is considered as a process of generalization in which the spatial context and attributes of objects from a source map are transformed into a target map. This is done according to a scale change through generalization operators such as selection, simplification, symbolization and classification (Robinson 1984; Shea and McMaster 1989). In remote sensing, however, thematic mapping is considered to be a process of pattern recognition in which the spectral responses of pixels are grouped into a number of defined classes using statistical modeling techniques. The process is also called image classification or pixel allocation (Burrough 1986). A problem with this process is that because of the complexity of the real world, spectral responses for a high resolution image show great spatial variability (Woodcock and Strahler 1988). From such heterogeneous data, it is difficult to directly generate homogeneous polygons, such as those presented in conventional maps. It is suggested that the concept of map generalization can be introduced to solve this problem.

It is well known that map generalization involves transformations in both the spatial domain and the thematic domain (McMaster 1989). In the spatial domain, map generalization refers to the transformation of points, line and polygons; in the thematic domain, it refers to attribute transformation. Traditional numerical generalization has focused on spatial transformation, specifically line generalization (Douglas and Peucker 1973; Shea and McMaster 1989; Muller 1990). This primarily involves a scale change.

In thematic mapping from remote sensing images, the key issue is the change in thematic representation. This may not involve procedures used in traditional map generalization, e.g., simplification. However, it could include other functions associated with raster representation, e.g., feature selection and feature smoothing (Monmonier 1983). At the same time, the process may not involve a scale change. We wish to argue that the classes obtained from image classification may not correspond to certain cartographic objects, as their spatial appearances are usually heterogeneous and their class membership may be uncertain (Robinson and Frank 1985). In this paper, we use the term ‘entity’ to describe the classes obtained from image classification. Map generalization, therefore, is concerned with the transformation of entities to cartographic objects.

In this paper, the aim is to extend the traditional map generalization concept into land use mapping from remote sensing imagery. To do this, a new procedure for mapping land use from satellite imagery has been devised. In this procedure it is assumed that land uses are highly generalized objects. As a result, they cannot be generalized directly from remote sensing imagery. The procedure has to be undertaken in two steps: image-to-entity generalization and entity-to-land-use generalization.

A CONCEPTUAL FRAMEWORK FOR MAPPING FROM IMAGERY

Map generalization is a complex mental process involving perception, cognition and other intellectual functions. "It focuses on the extraction of the general, crucial elements of reality" (Brassel and Weibel 1988, p.230). It is usually related to the functions of selection, simplification, emphasis, classification, etc., by which observed reality is structured into a number of individual entities; then important entities are selected and represented on the map. Brassel and Weibel (1988) proposed a five-step conceptual framework for map generalization:

- *Structure Recognition* aims at the identification of objects, understanding their spatial relations and the establishment of measures of relative importance. It is the basic understanding of the essential structures of the spatial information available in the original database.
- *Process Recognition* is to establish the relationships between source objects and generalized objects (e.g., linguistic relations, spatial relations and statistical relations), based on the structure of the original database and the control parameters (e.g., objective).
- *Process Modeling* can be considered as a compilation of rules and procedures derived from a process library and the pre-setting of the process parameters that were established during process recognition.
- *Process Execution* and *Data Display* are operational procedures (e.g., classification, simplification and symbolization) which convert database and information structures into the target and generalized databases. These

procedures have been addressed in many existing approaches (McMaster 1989; Steward 1974).

Mapping from imagery is a generalization, which represents a process of transformation from the digital (spectral) domain to thematic and spatial domains. This process, however, is different from conventional cartographic generalization because the spatial units for generalization are not cartographic objects. In other words, they are not points, lines and polygons, but rather they are pixels which do not have any thematic meaning.

A two-stage procedure for mapping from imagery can be structured: statistical generalization and cartographic generalization. In statistical generalization, the original imagery is divided into a number of entities derived under statistical control. This represents processes of data reduction and transformation. The result is not a map but an entity image which shows basic spatial structures and thematic components of the remote sensing imagery. In cartographic generalization, the object is highly generalized, selective and subjective. Relationships between information entities and cartographic objects are modeled to produce a smooth, uniform map. A conceptual framework for mapping from remote sensing imagery is shown in Figure 1.

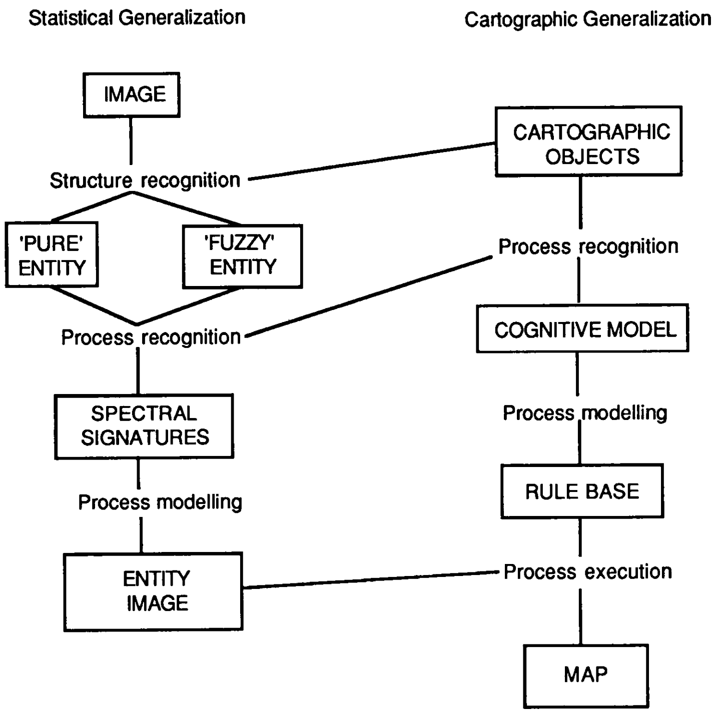


Figure 1 A conceptual framework for mapping from imagery.

In statistical generalization, the imagery should first be examined according to the mapping objectives. From this examination, a list of entities should be prepared. There are two types of entity: 'pure' entity and 'fuzzy' entity. A 'pure' entity is one which is homogeneous and can be clearly identified on the image, while a 'fuzzy' entity consists of mixtures of 'pure' entities and is ambiguous when observed on the image. The original image can be transformed into an entity image containing the two types of entity. These entities constitute the basic information for a cognitive model in cartographic generalization. In the process modeling stage, sampling procedures can be employed to extract spectral signatures in order to link the spectral values of the image with entities. In the process execution stage, a classifier can be used to assign each image pixel to an entity label, based on the results of the sampling.

In cartographic generalization, the relationships between entities and cartographic objects (such as logical relation, spatial relation and statistical relation) need to be identified in order to develop a cognitive model. Based on this model, a rule base for the generalization can be designed. The rule base may be constructed as a simple logical operation or as a more sophisticated expert system, depending on the complexity of the mapping task. In the final stage, process execution, a map is generalized from the input entity image.

A CASE STUDY

Based on the conceptual framework presented above, a case study of land-use mapping from SPOT imagery was carried out. The land-use mapping procedure was divided into two steps: entity extraction and land-use map generation.

Study Area and Data Description

The test site selected for the study is part of the city of Scarborough, one of the fastest-growing municipalities in Metropolitan Toronto, Canada. The study area is dominated by residential areas at different stages of development; industrial and commercial land uses are also prevalent. The image used for study was a subscene (256 x 256 pixels) from a multispectral SPOT image with 20 m x 20 m spatial resolution pixels. It was acquired on June 4, 1987 (Figure 2).

Entities and Land Uses Identifiable from the Image

Based on the generalization concept, two types of entity in the image, the 'pure' entity and the 'fuzzy' entity, were identified. 'Pure' entities have a distinct spectral appearance on the image and have relatively narrow spectral distributions (i.e., the digital values have relatively low standard deviations). A 'fuzzy' entity is not defined precisely, but it has a relatively wide range of spectral values. In Table 1, eight entities which were recognized on the image are listed. Asphalt surface, concrete surface, bare surface, and the two types of trees are 'pure' entities; the other three are 'fuzzy' entities.

The objective of the case study was to map land use from the image. Six land uses were categorized (Table 2). As can be seen, each land use is composed of several land-cover entities (Campbell 1983). It should also be noted that the relationships between land-use types and entities are quite complicated and that many land uses have similar entity components.

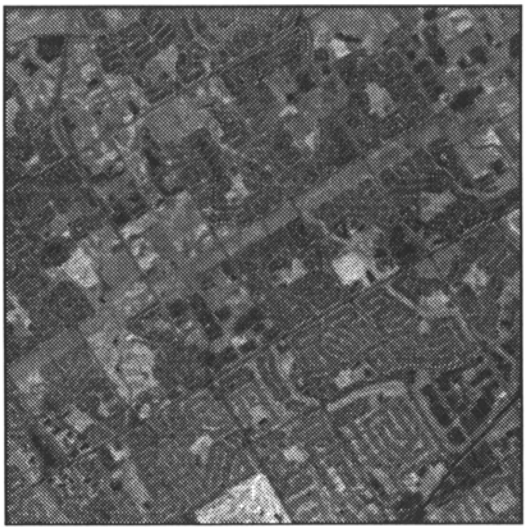


Figure 2 A SPOT image of northern Scarborough, Ontario, Canada recorded on June 4, 1987.

Modeling Generalization from Entity Image to Land-Use Map

Modeling the generalization from entity image to land-use map involves establishing relationships between entities and cartographic

Table 1. Entities and Their Descriptions

Code	Entity	Description	Gray Level
1	Asphalt surface	roads, house roof, parking lot	1
2	Concrete surface	building, warehouse, parking lot	2
3	Bare surface	land cleared for construction	3
4	Soil surface	wasteland, non-cultivated surface	4
5	Deciduous trees	deciduous trees	5
6	Coniferous trees	coniferous trees	6
7	Low-density grass	grassland, lawn	7
8	High-density grass	grassland, lawn	8

Table 2. Land Uses and Entities Contributing to Them

Code	Land Use	Composition*	Gray Level
A	Old residential	1, 2, 4, 5, 6, 7, 8	1
B	New residential	1, 2, 4, 7, 8	2
C	Industrial/Commercial	1, 2, 4, 5, 6, 7, 8	3
D	Land under construction	1, 2, 3, 4, 7, 8	4
E	Open space	4, 7, 8	5
F	Woodland	5, 6	6

*Numbers represent codes in Table 1

objects (i.e., building a cognitive model). However, the relationships between entities and land uses are complex; there is no one-to-one correspondence between them. It is, therefore, important to select the key factors or parameters to model the generalization. A simple arithmetic aggregation of entities into a specific land-use type, such as land use A = entity 1 + entity 3 + entity 5, is insufficient to accomplish the generalization process from entities to land use. Fortunately, the proportional distributions of different entities vary from one land use to another. Therefore, it is possible to model the generalization process from entity-image to land-use map using the spatial frequency of each entity as a parameter (Gong 1990). For example, high-density grass (entity 8) has contributions to both residential (land use A and land use B) and open space (land use E), but the spatial frequency of high-density grass in open space is much higher than in residential areas. Based on the different spatial frequencies and entity compositions, the various land uses can be distinguished.

Entity-Image Generation

The procedure used to produce the entity image was a supervised maximum-likelihood classification. First, spectral signatures of the eight entities were obtained using a supervised training algorithm. The entire image was then classified according to these spectral signatures using a maximum-likelihood classifier. Figure 3 shows the entity image obtained using this method; gray levels are listed in Table 1.

Generalization from Entity Image to Land-Use Map

Two procedures were employed to carry out the generalization from entity image to land-use map. The first procedure, based on differentiating entity frequencies, was used to derive old residential (land use A), new residential (land use B), industrial and commercial (land use C) and land under construction (land use D). The second procedure, arithmetic aggregation, was used to extract open space (land use E) and woodlot (land use F).

In the entity-frequency-based procedure, a pixel window (9 x 9) was first moved over the entity image to extract an entity-frequency

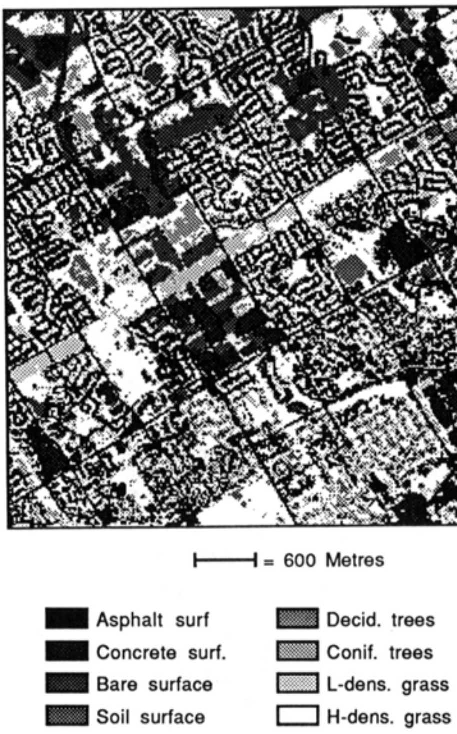


Figure 3 An entity image generalized from the SPOT image.

vector $F(i,j) = \{f_1(i,j), f_2(i,j), ..., f_8(i,j)\}^T$. $F(i,j)$ was associated with the center pixel of each pixel window at row i column j on the image. $0 \leq f_k(i,j) \leq 81$ $(k = 1, 2, ..., 8)$ denotes the occurrence frequency of entity k in a pixel window. To determine whether pixel (i,j) belonged to one of the land-use types A - D, a city-block distance measure was used:

$$d_m(i,j) = \sum_{k=1}^{8} \left| f_k(i,j) - c_{mk} \right|$$

where $d_m(i,j)$ is the distance from the entity frequencies at pixel (i,j) to the average entity frequency $c_m = \{c_{m1}, c_{m2}, ..., c_{m8}\}^T$ for land use m $(m = A, B, C, D)$; c_m was obtained from supervised training on the entity image.

Once $d_m(i,j)$ was obtained, it was compared with a threshold ß (0<ß<81). If $d_m(i,j) < \beta$, pixel (i,j) was a candidate for land use m. Otherwise, pixel (i,j) was rejected from land use m. If more than one land use was a candidate, pixel (i,j) belonged to the land-use type for which the distance was the shortest. A detailed description of entity-frequency extraction and land-use identification based on entity frequencies can be found in Wharton (1982), Zhang *et al.* (1988) and Gong (1990).

In arithmetic aggregation, two aggregation rules were used:

land use E = entity 4 + entity 7 + entity 8

land use F = entity 5 + entity 6.

However, entities 4, 5, 6, 7, and 8 are also components of land uses A - D. Therefore, a conflict will arise when an entity label at pixel *(i,j)* belongs to one of the land uses A - D and one of the land uses E - F. Under such circumstance, pixel (i,j) was assigned to one of the land uses A - D. This is reasonable because in the first procedure both entity information from pixel *(i,j)* and neighborhood entity information from a pixel window were included in the identification.

After the two procedures, a number of pixels remained unlabeled. These unlabeled pixels were relabeled using the entity-frequency-based method, but without thresholding the distances $d_m(i,j)$.

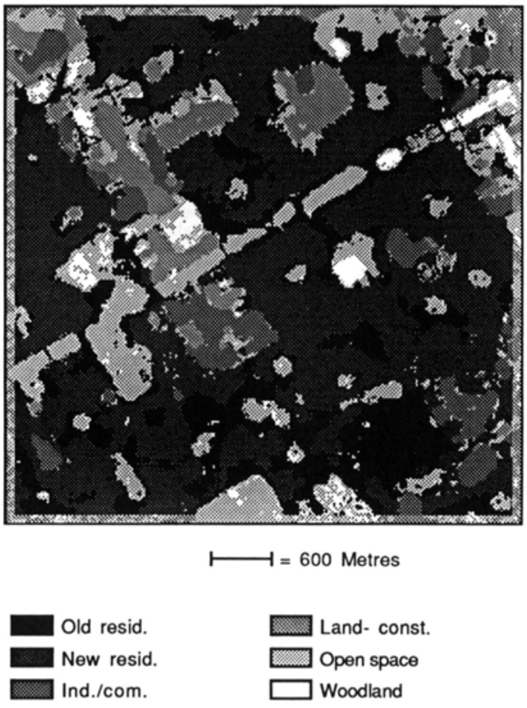


Figure 4 A land-use map generalized from the entity image.

Results and Discussion

The map in Figure 4 is a result from this preliminary study. It shows homogeneity in polygon distribution, which is consistent with a conventional cartographic product. By visual comparison, most land uses on the map have corresponding locations on the image. The results show the potential for mapping from remote sensing imagery using the generalization concept. However, there are still some problems to be overcome:

- The old residential land use is confused with open space, because there is high spatial frequency of grass cover in the old residential area.
- There are still some "salt-and-pepper" patterns on the map; smoothing is required.
- The selection of land uses is restricted. Some land uses, which can be identified by visual interpretation (e.g., recreational land use), are not generalized.

These problems result from modeling during the generalization. In modeling, only component factors were considered, while in human perception, spatial features such as shape, size, linearity and spatial adjacency are also important. It may not be possible to represent these factors by a statistical model; a fuzzy model or a logical model may be more appropriate. It is apparent, however, that more sophisticated models are required in the entity-to-map generalization procedure, such as an expert system. In future studies of entity-to-map generalization, a fuzzy-set-theory approach and a knowledge-based approach should also be considered.

CONCLUSIONS

It is concluded that thematic mapping from remote sensing data is a challenging issue in numerical map generalization. It involves a process of thematic information extraction and entity-to-cartographic-object generalization, during which a scale change may not be involved. Research on this topic is limited. In this paper, we proposed a conceptual framework for mapping from imagery, based on the generalization concept. The aim of this approach was to extend the conventional cartographic generalization concept to remote sensing data, and to rethink the theoretical foundations for mapping from remote sensing. A case study of land-use mapping was undertaken to verify the theoretical model. A homogeneous land-use map was presented as a preliminary result for this procedure. Although the result is still not as good as that in human perception, it demonstrates the potential of the new methodology for mapping from remote sensing imagery. Further work involving more sophisticated models is justified.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the assistance of SPOT Image Corporation of France and the Canada Centre for Remote Sensing in supplying the SPOT data used in this study. This research is funded by a Centre of Excellence grant from the Province of Ontario to the Institute for Space and Terrestrial Science, and NSERC Operating Grant A0766 awarded to Dr. P. J. Howarth. Thanks to Denis Gratton for assistance with preparing the figures.

REFERENCES

Brassel, K. E. and R. Weibel 1988, A review and conceptual framework of automated map generalization, International Journal of Geographic Information Systems, Vol. 2, No. 3, pp. 229-244.

Burrough, P. A. 1986, Principles of Geographical Information Systems for Land Resources Assessment, Clarendon Press, Oxford.

Campbell, J. B. 1983, Mapping The Land -- Aerial Imagery for Land Use Information, Resource Publications in Geography, The Association of American Geographers, Washington, D. C.

Douglas, D. H. and T. K. Peucker 1973, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, The Canadian Cartographer, Vol. 10, No. 2, pp. 112-123.

Gong, P. 1990, Improving Accuracies in Land-Use Classification with High Spatial Resolution Satellite Data: a Contextual Classification Approach, Unpubl. Ph.D Thesis, Department of Geography, University of Waterloo.

McMaster, R. B. 1989, Introduction to 'Numerical Generalization in Cartography', Cartographica, Vol. 26, No. 1, pp. 1-6.

Monmonier, M. S. 1983, Raster-mode area generalization for land use and land cover maps, Cartographica, Vol. 20, No. 4, pp. 65-91.

Muller, J. C. 1990, The removal of spatial conflicts in line generalization, Cartography and Geographic Information Systems, Vol. 17, No. 2, pp. 141-149.

Robinson, A. H. 1984, Elements of Cartography, John Wiley and Sons, Inc., New York.

Robinson, V. B. and A. U. Frank 1985, About different kinds of uncertainty in collections of spatial data, AutoCarto 5, Washington, D.C. pp 440-449.

Shea, K. S. and R. B. McMaster 1989, Cartographic generalization in a digital environment: when and how to generalize, AutoCarto 9, Baltimore, MD, pp. 56-67.

Steward, H. J. 1974, Cartographic generalization: some concepts and explanation, Cartographica Monograph, No. 10.

Wharton, S. W. 1982, A context-based land-use classification algorithm for high-resolution remotely sensed data, Journal of Applied Photographic Engineering, Vol. 8, No. 1, pp. 46-50.

Woodcock, C. E. and A. H. Strahler 1987, The factor of scale in remote sensing, Remote Sensing of Environment, Vol. 21, pp. 311-332.

Zhang, Z., H. Shimoda, K. Fukue, and T. Sakata 1988, A new spatial classification algorithm for high ground resolution images, Proceedings of IGARSS'88, Edinburgh, Scotland, pp. 509-512.

PRODUCING ANSWERS TO SPATIAL QUESTIONS

Gail Langran
Intergraph Corporation
2051 Mercator Drive
Reston, VA 22091

ABSTRACT

This discussion explores alternatives to standard GIS command procedures. The goal is for the user to describe the information he or she seeks rather than the data manipulations that should be performed so the system can provide appropriate display content and format. If the same information model were used for both user input and display generation, then spatial, temporal, or thematic questions could be matched to tabular or graphic answers. Using a model of geographic information, the potential for an artificial language that would permit a user to phrase geographic questions using English-like grammar and language is examined.

INTRODUCTION

Since their inception, GISs have become increasingly more sophisticated in terms of standardization, data structuring, error control, and analytical options. However, ease of use is still a major obstacle to the full exploitation of GIS technology. Many systems force new users to enter the world of GIS by navigating a maze of command-line interfaces, voluminous documentation, and user-built displays. This initial investment in learning a new tool may be beyond the means of busy analysts with other options for performing their analyses. It also speaks ill of the GIS discipline; if the main responsibility of an information system is communication of information, it follows that no mattter how sophisticated the storage and analytical capabilities, these systems somehow fail unless communicative powers are developed (Webber 1986b).

The harshness of the GIS user environment is gradually softening. On-line help sequences and self-explanatory point-and-click input forms are changing the book-on-the-knees posture so common to users of command-line GISs. Special-purpose systems that address the specific needs of a particular application (for example, oil exploration) tend to be simpler to use because they are geared to users whose expertise is expected to be in areas other than GIS. But the difficulty of formulating a set of commands in a multi-purpose GIS and the tedium of selecting an appropriate display format for responding to queries to the system remains.

Special-purpose cartographic query languages do exist. Nyerges (1980) developed a query language geared specifically to cartographic purposes that permitted users to request information via a grammar and keywords that the system could decode. Frank (1980) developed a query language that could automatically produce a display of the data selected if relational logic alone were used. Broekhuysen and Dutton (1983) describe the design of the Odyssey command language and their efforts to achieve the effect of a dialog with the computer. Morse (1987) describes an expert system for forest resource management that accepts if-then statements from a user, translates those into GIS commands, and produces a standard report.

Shortcomings of the present approach
A universal problem among current approaches is the intuitive barriers that users face when combining relational and spatial logic to describe a course of action to the computer. Egenhofer et al. (1989) have addressed this problem by augmenting the query language style with "point and click" methods of indicating the objects involved. While this approach will ease the plight of the user, it does not change the fundamental fact that so-called "query languages" are essentially ways of defining a subset of data to retrieve without indicating the purpose of the retrieval. In contrast, the ideal mode of discourse with a GIS is for a user to describe what information is sought rather than how the system should manipulate the data to produce the information. Compare a high-level programming language to assembly language.

High-level: *Add 2 + 3 and store the results in A.*

Assembler: *Place the values 2 and 3 in registers, add the registers, place the result in another register, whose number the user must track for subsequent manipulations of the result.*

Today's GISs users express their information needs in the semantic equivalent of assembly language. A preferable mode of discourse with the system would provide the user with a means to express his or her information needs, as opposed to data-manipulation steps. Compare the following dialogs.

Ideal: *Tell me where cow pastures border this stream.*

Current: *Build a narrow buffer around the stream, overlay the buffer with the data, select cow pastures from within the buffer, create a map whose extent covers the stream, shade the agricultural areas in a selected pattern or tint.*

The fact that the logic of data retrieval and spatial overlay is slippery to many GIS users does not imply that GIS users are slow when compared to users of other information systems. Rather, the methods we presently use to interact with computers are designed to be straightforward to computers rather than to humans. Katzeff (1989) tested novice database users on their ability to construct a query and predict a response. Most of the users did correctly predict the response, but only one-fifth were able to construct a correct query. A question and a desired answer may be clearly in mind, but translating that question into a command sequence that produces the desired answer is a challenge.

Given that questions and answers are clear, while the precise data manipulations needed to obtain an answer are not, it is worthwhile to investigate whether an information system could be designed to receive "raw" questions and produce comprehendable answers without involving the user in the specification of data manipulations. Information needs are easily expressed as questions to the system. Where are all the slopes steeper than 10%? Who owns this parcel? When did this parcel last change hands and who was its previous owner? If a user could inform a system of the information being sought, the system would have the raw materials to provide more helpful help sequences, on-line documentation, and default displays. A move toward this ideal requires two major enhancements to the current approach: users need a more straightforward way to express their information needs and GISs need an automated method of choosing a display format.

Natural language is one route to facilitating human/computer dialogs. But practical considerations demand that we proceed to investigate a higher level of human-machine discourse without waiting for natural language processors, since work in that area is still in its infancy (see Quillian 1985, Schank and Rieger 1985, and Webber 1986a for surveys of the natural language approach; see Morse 1987 and McGranaghan 1989 for geographic applications of natural language). A simple artificial language that operates at a similarly high semantic level could produce results comparable to a natural language, since it appears that syntactical constraints do not impede users if the underlying logic of the discourse is clear. Borenstein (1986) compared the learning abilities of users equipped with natural-language help sequences to those who used a verb-noun artificial command language and found no significant difference. Thus, this work investigates the design of an artificial language to describe geographic questions.

The next section describes the problem in more detail and is followed by a presentation of a theoretical basis for designing a high-level question-answer mode of discourse for users of GISs. Later sections examine the elements of an artificial language to express geographic questions in such a way that the computer can answer with default displays, and summarize the goals and findings of this study.

STUDY GOALS

A high-level method of human/computer dialog that describes information needs rather than data-processing instructions is evidently desirable. The issues that must be addressed for such an improvement to become a reality are

What syntax would the artificial language use?

How would questions be linked to data-manipulation commands?

How would the system choose appropriate display formats?

The first and last of the three issues are open questions. The second issue, while equally challenging, has been treated to some degree. Wu et al. (1989)

describe a frame-based GIS that can receive an expression in a high-level formal language, translate the expression into primitive data manipulation procedures, order the primitives by predefined optimization rules, and execute the commands. The query language developed by Nyerges (1980) also had a multi-level structure comprised of a query language, query decoder, and query processor. While enhancements are always useful, the work of Wu and Nyerges demonstrates the premise that GISs can be informed with sufficient intelligence to match high-level commands to low-level procedures (Table 1).

Table 1. Producing answers to geographic questions. An event sequence includes intermediate dialog between the human and the computer to verify the treatment of a question.

Human	**Computer**
Frame a question or questions.	Match the questions to a set of commands.
Verify the command structure.	Propose a default display format.
Verify display format.	Perform requested data manipulations.
	Produce display format.

To address the questions of syntax and display selection, it is useful to form some preliminary requirements. The query syntax should implicitly embed the information needed to select the appropriate content and format of a response display. If that were the case, standardized answers to questions could be produced in a range of formats without burdening a user with cartographic decisions, either at the micro (e.g., color or gray scale, pattern, shape, generalization, font) or macro (e.g., map type, geographic window, scale selection, entities depicted) level. Several interesting attempts have been made to automate the type of decisions referred to here as "micro" (see Robinson and Jackson 1985, Muller 1986, Mackeness 1987, and Weibel and Buttenfield 1988). It is the "macro" decisions that remain unaddressed. Macro decisions are linked less to legibility than to semantic integrity; in other words, a person may respond to a question in well-modulated tones and correct grammar, but if the answer is off target, the effort is in vain.

If questions are straightforward (e.g., "Where are all the forests in this area?") a display is relatively easy to produce automatically. Frank (1982) describes a GIS query system that could match simple queries with displays by automatically determining a window and scale, then selecting the requested relational entities that would appear. But as demands on GISs become more intricate and GISs themselves become more complex (e.g., by incorporating temporal information), the selection of default displays becomes correspondingly more complex. Entities may be mentioned in a question that are not included in the display that answers it. More complex questions require more complex syntax and a broader system vocabulary. And many output formats become possible and needed (Table 2).

Table 2. Different questions require different formats for answering.

Question	**Format**	**Content**
Where are the pastures in this watershed?	Map	Highlight pastures
What are the different types of agricultural land use in this watershed?	Map Listing	Color or list by agricultural type
Who owns the pastures that border the stream?	Map Listing	Color or list by owner
What is the grazing density of the pastures beside the stream?	Map Listing	Shade or list by density
Where does the stream border pastures?	Map	Highlight stream segments
What pastures have changed from nonagricultural use?	Map Listing	Highlight or list pastures
When did they change?	Map Listing	Shade or list by years
How many acres changed?	Value	Number of acres

The problem of default displays becomes more complex when a GIS includes temporal data, since, in addition to spanning space, the queries can span time and space/time. For example, the simple question of where a given feature or attribute type occurs becomes more complex when the question of occurrence concerns a time in the past (i.e., where it occurred ten years ago); a timespan (i.e., where it has occurred any time during the past ten years); flows, motion, or trends (which imply a timespan when the movement occurred); or a change over time (i.e., where it has changed from one feature or attribute to another over a given timespan). Vasiliev (1990) discusses different forms of temporal maps and provides examples of the many methods available for expressing spatial change in graphic terms.

Current GISs place the onus for specifying how the reply should look on the user (Figure 1). Will it be graphic or tabular? If graphic, will it be an outline map dotted with point symbols or a choropleth map? What colors, shapes, and patterns should be used? If the display is tabular, what are the rows and columns? How wide should they be? Ideally, the next generation of GISs will shoulder this responsibility unless a user specifically asks to share it.

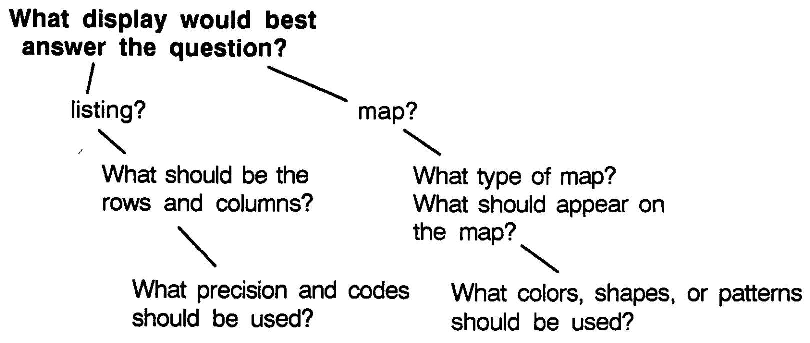

What display would best answer the question?
listing?
map?
What should be the rows and columns?
What type of map? What should appear on the map?
What precision and codes should be used?
What colors, shapes, or patterns should be used?

Figure 1. Display options for answering a question posed to a spatial database.

THEORETICAL BASIS

Evidently, some mechanism must exist for mapping questions to formats for answers. Two general approaches are possible: enumerate high-level information requests and map each to a default display, or develop a classification scheme for questions and answers and a means of recognizing what class of question has been posed.

Enumerating GIS operations

Many different attempts have been made to enumerate GIS operations. The capabilities that different authors describe as useful to a GIS can be divided into two groups: information desired, and functions available. Table 3 is an aggregated listing of these two classes that was collected from Nystuen (1968), Salmen (1977), Honeycutt et al. (1980), White (1984), Wu et al. (1989), and Guptill (1989).

Table 3. Types of GIS information and functions.

information desired	**functions available**
multi-scale analysis	windowing
multi-map compositing	rotate, shift, scale
spatial clustering and aggregation	attribute aggregation
edge detection	map overlay - union
direction of flow	map overlay - intersection
comparison	map overlay - negation
precedence	calculate area, length, volume
coincidence	calculate aximuth, bearing, coordinates
proximity	calculate statistics from tabular data
adjacency	buffer zoning: erode and dilate
interpolation	search (locate all)
corridor delineation	line smoothing or simplification
slope and aspect	point in polygon
optimum path	point in line
feature recognition from geometry	line in polygon
weighting	
intervisibility	

The problems of enumeration become apparent when one investigates the enumerations that exist: no two enumerators have arrived at identical listings, the completeness of any given enumerations is debatable, and none include a temporal dimension. Given the immaturity of GISs in general, and temporal GISs in particular, few would argue that any enumeration of information needs could be considered exhaustive. A question-to-answer mapping built on this approach would need to be adjusted or expanded continually.

Classification of operations

The alternative to "hardwiring" questions to answers is to adopt a classification scheme for both that permits questions to be mapped to answers of the same class. Logically, a classification scheme must be built on a reasoned understanding of the information involved. This follows the thinking of Booth (1989) and Jarke and Vassiliou (1985), who argue that establishing a mutual conceptual framework aids in arriving at a mutual understanding of the topics being discussed. Even in the very restrictive setting of a human/computer dialog, a common view of the information being treated seems fundamental. Various authors offer conceptual frameworks concerning the nature of geographic information and operations. None was designed to be used as a semantic basis for a human-computer dialog; however, each has merit.

The first framework considered was that of Tomlin (1983), who names three types of geographical modeling operations.

- The output value is a function of a point.
- The output value is a function of neighborhood or adjacency. Neighborhoods can be immediate, extended, or indeterminate (i.e., the neighborhood must be computed or estimated after the process is underway)
- The output value is a function of a vicinity or region.

These three types of operations would be multiplied in a temporal database, since each type of data could exist at a point in time or in a trajectory through time, and each modeling operation could consider a point in time or a trajectory through time.

A second prospective framework is that of Rucker (1987), who describes a mathematical treatment of reality that could serve as a basis for a model of geographic information. As stated by Rucker, the five archetypical patterns of mathematics are number, space, logic, infinity, and information. A geographical region can be used to illustrate these concepts.

- Number. A region contains a certain number of buildings and a certain number of wetlands. The buildings have a certain height and length that can be measured numerically; the wetlands have measurable moisture, animal populations, and acreage. The area of the region itself can be measured, as can its population.

- Space. A region is not flat, as it might appear on a map. It exists in four-dimensional space/time. It has no holes (as defined here), and it connects to other regions. It follows the curvature of the earth, its stream network branches in a one-dimensional pattern, and the earth's relief forms roughly conical bulges in three dimensions. The region has subregions, which might by accessed by White's (1984) windowing, buffering, boundary, and endpoint operators. Subregions also may intersect, coincide, or be included within one another.

- Logic. The subregions within a region are interconnected. The region also reacts to external changes. A dammed stream floods land upstream. A zoning change alters land use. Tax disparities between regions cause outmigration or unpredictable settlement. Interconnections and reactions also exist that are unknown or poorly understood.

- Infinity. By zooming out from the region, one might see that it forms a pattern with other regions. By zooming in, one might notice greater detail and apparent structure within that detail and cells, and then atoms come into focus. What meets the eye when examining a region relates closely to the scale at which the region is examined.

- Information. Over time, a region is subject to random influences that leave their mark. Rucker suggests two ways to measure the information content of an entity: by the number of questions required to build a replica of the entity, or as the length of the shortest computer program required to answer any possible question about it.

As one might expect, this mathematical model addresses conceptual units and measurements, which could be useful in selecting map formats. However, it is not linked to the components of geographic information and hence would need to be extended considerably to meet the needs expressed in this discussion.

A more promising framework is that of Sinton, whose 1978 work on spatial data representation is useful for organizing spatiotemporal information because it addresses all three components of spatial information: attribute, location, and time. Sinton argues that traditional representation methods can measure only one of these attributes. A second is fixed to a constant value, and the third is controlled to a range of values or a set of categories (Table 4).

Sinton starts with a map and classes it according to how the map treats the various components. If it were possible to start with a question and class it according to how the question treated each component, then Sinton's framework provides a likely starting point for addressing issues one and three because questions could be matched to the appropriate formats for answers.

Table 4. The representation of geographic data in various formats (extended from Sinton 1978).

	Fixed	**Controlled**	**Measured**
Soils data	time	attribute	location
Topographic map	time	attribute	location
Census data	time	location	attribute
Raster imagery	time	location	attribute
Weather station reports	location	time	attribute
Flood tables	location	time	attribute
Tide tables	location	attribute	time
Airline schedules	location	attribute	time
Moving objects	attribute	location	time

Sinton's framework revolves around the components of the information itself, rather than functions or measures alone. In addition, the framework is tied naturally to graphic forms, so an artificial language that is structured to express what components are fixed, controlled, or measured could also indicate what output form is appropriate. Returning to the goal of a dialog based on questions and answers, we can see that Sinton's framework can be mapped readily to question words.

Attributes:	what, who, how many, how much
Location:	where
Time:	when, how long

Using that mapping, a tie between question and answer is already evident, since the "question word" indicates the measured variable.

AN ARTIFICIAL LANGUAGE FOR POSING GEOGRAPHIC QUESTIONS

A high-level artificial language built upon questions designed to elicit answers would alleviate the difficulty of using a GIS. Today's GISs force the user at the helm to express information needs using a combination of relational and spatial logic. Questions are not asked directly, so the system has few options for providing meaningful help sequences or display formats.

How it would work

The high-level language would be designed to sit atop of the attribute and spatial command languages that dictate data manipulations. High-level commands would call lower-level commands, just as a high-level programming language decomposes to machine language before functions are performed. This concept is similar to the methods adopted by Nyerges (1980) and Wu et al. (1989), although in those two implementations, the top-level query language did not disassociate the user from data-manipulation commands, as supported here. As demonstrated by Wu et al., the effect of the high-level layer on performance is negligible; the translation of question-to-command occurs before commands are executed, and a language can be designed for direct translation into commands.

The high-level language would be comprised of a limited set of verbs, nouns, and modifiers and a grammar to describe sequence within an expression. The system would parse the expression and select the appropriate data retrieval and manipulation commands to execute. At the same time, the sequence in which entities are mentioned would indicate which elements of the expression were fixed, controlled, or measured. That information, and identifying whether the data types are point, line, or area, would indicate what output format to select.

Although this study has not defined a query language that meets the goals listed above, certain patterns in the sequence of words in questions indicate a possible syntax. It is useful to note the natural syntax of English questions to ensure that an artificial language is truly "English-like" and easy to learn. Using the questions from Table 2 (above), Table 5 describes how natural English grammar contains clues concerning measured components, content desired, and window.

Table 5. Common English grammar used to frame questions gives a basic indication of the format required to answer. Questions are taken from the examples of Table 2. In all cases, the region of interest in "this watershed."

Measured component	**Primary subject**	**Attribute modifier**	**Relative location**	**Period of interest**
Where are	pastures			now
What is	agriculture	by type		now
Who is the	owner	of pastures	by stream	now
What is	grazing density	of pastures	by stream	now
Where does	stream		border pastures	now
What	pastures	changed from nonagriculture		past ten years
When did	pastures	change from nonagriculture		ever
How many	acres	changed from agriculture to nonagriculture		past ten years

To map from a question to an output format is not entirely straightforward, even when the natural-language version appears so easily parsed. Aside from the question "where...," which is answered most naturally via a map, most "question words" can be answered in either map or tabular format. Accordingly, a user should be able to select either a map or a listing as output unless one is patently inappropriate. Ideally, the system can recognize when only one ouput format will do. Use of the word "where" is one clue. Others also exist; the single question listed in Table 5 that requires a listing has a unit of measurement as its primary subject.

A second problem of selecting a map design by default is informing the computer of the data types involved. Attributes can use nominal, ordinal, interval, or ratio measures, and the mapped entities can be points, lines, areas, or surfaces. Each combination has a set of appropriate mapping techniques. The logic required to select an output format would follow the lines offered in Table 6. Once again, criteria for each decision will be collected from different parts of the question. The question word combined with the subject indicates the level of measurement. For example, a "where" question requires only nominal symbols to answer, but a "what" can require an ordinal, interval, or ratio answer depending on the subject. *What is the agriculture by type?* requires a different map response than *What is the grazing density?* A reasonable approach to automating this decision is to include the level of measurement for each data attribute in the data dictionary.

Table 6. Deducing an appropriate output format given a question to answer.

Select entities to appear on the map using the region and period of interest, the primary subject and attribute modifiers, and constraints on relative location.
Determine whether the primary subject is point, line, or area.
Determine whether the measure is nominal, ordinal, interval, or ratio.
Select format.

Although Table 5 decomposes the questions only to a coarse level of detail, more information exists in the natural expressions that help describe the desired manipulations and output format to the computer. For example, the verbs include tense, which indicate a temporal map and an excursion into the past-tense database. In addition, certain words (e.g., "border..." and "change...") are keywords that describe a buffering and a temporal operation, respectively.

Possible enhancements

Several measures are possible to move the dialog to yet a higher level of discourse. Specialized views or objects could be developed to express to the system combinations of data that have special meaning to the user. Using Table 5's example, the watershed could be stored as a named MBR in the database so it could be referenced by name and a window automatically selected. Alternately, a user could indicate the region of interest on the screen, as described in Egenhofer et al. (1989).

Other enhancements to a question-oriented query language could include using expert system techniques to shorten the instructions necessary to describe certain concepts to the computer. Two excellent examples of "mini" expert systems are found in the temporal information system literature. Overmyer and Stonebraker (1985) describe the development of a "time expert" within a relational database that permits the system to interpret such terms as "lunch," "today," incomplete dates, and ranges of dates. Kahn and Gorry (1977) describe a "time specialist" that interprets temporal constructs needed in problem solving. The Overmyer and Stonebraker system operates within an INGRES QUEL environment, while Kahn and Gorry's system is designed to interpret natural language expressions. Input method could equally well be query-by-example, graphics, or a fourth-generation language.

Additional bonuses

The focal purpose of developing a question-oriented query system is to relieve users of the task of specifying data-manipulation procedures to the computer, and to permit the computer to produce default displays automatically in response. The latter in particular is most effective if the computer understands the information sought.

If the human-computer dialog informed the computer of what information is sought by the user, conceivably the system could supply more useful help sequences. These would include model questions that the user could choose from and edit, and listings from the data dictionary. Incorrect syntax could be corrected through gentle feedback from the computer regarding correct options and the results they would produce.

Assuming that the high-level query language were built on the georelational architectures in common use today, the high-level question would be decomposed by the system into a series of SQL and spatial commands. One option for experienced users would be to list these commands and permit them to be edited for greater control over output. New users also could employ such a listing as a learning tool.

SUMMARY AND CONCLUSIONS

This discussion does not introduce a finished high-level query language. It does, however, supply a basis for continuing investigation into the topic. It argues the following.

- The next generation of GISs should relieve users of the burden of specifying data manipulations and output formats by permitting them to specify their information needs (rather than data-processing needs) in an English-like artificial language supplemented by point-and-click inputs.

- The design of any cartographic query language should be linked to the problem of how a query's response should be formatted. At present, systems permit users to design and save an output format, and reference that format as the desired output for a query. Arguably, this level of automation can be increased and system designers can relieve users of the burden of map specification altogether.

- Sinton's theory of geographic representation provides a possible starting point for developing a conceptual framework for questions and answers that the human and computer can share.

Continuing research in this area can take two parallel tracks by attacking the problem from its two ends: the query language and the output format. Development of a query language requires an investigation of the natural phrasing of geographic questions and how their elements decompose and map to data manipulations, and further examination of how Sinton's theory applies to geographic questions. Development of default output formats requires a better understanding of what formats best answer what questions, and how the components of the question indicate the composition of the format.

Many have argued for the importance of cartographers in the GIS discipline. Given the current state of human/GIS communication, however, one might wonder at how involved cartographers have been to date; the cartographer's purported interest in communication has not apparently improved the state of affairs for GIS users. Until now, much of the harshness of the GIS environment could be attributed to the immaturity of hardware and software systems. But the raw materials do exist now to better the lot of GIS users considerably; it is our understanding of geographic questions and answers that lags behind.

REFERENCES

Biskuo, Joachim; Räsch, Uwe; and Stieteling, Holger. 1990. "An Extension of SQL for Querying Graph Relations." *Computer Languages* 15, 2, 65-82.

Booth, Paul. 1989. *An Introduction to Human-Computer Interaction.* East Sussex: Lawrence Erlbaum Associates Ltd.

Borenstein, Nathaniel S. 1986. "Is English a Natural Language?" *Foundation for Human-Computer Communication,* K. Hopper and I. A. Newman, editors. North-Holland: Elsevier Science Publishers BV, 60-72.

Broekhuysen, Martin and Dutton, Geoffrey. 1983. "Conversations with Odyssey." *Proceedings of Auto-Carto 6,* Ottawa, November, Vol. 2, 15-24.

Egenhofer, Max J.; Frank, Andrew U.; and Hudson, Douglas L. 1989. "User Interfaces for Geographic Information Systems." Abstracts of the 14th World Conference of the International Cartographic Association, August, Budapest, 162.

Embley, David W. 1989. "NFQL: The Natural Forms Query Language." *ACM Transactions on Database Systems* 14, 2, 168-211.

Frank, André. 1982. "MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and their Graphical Representation." *Computer Graphics* 16, 3, July, 199-207.

Guptill, Stephen C., editor. 1988. "A Process for Evaluating Geographic Information Systems." Federal Interagency Coordinating Committee on Digital Cartography, U.S. Geological Survey Open-File Report 88-105.

Honeycutt, Dale; Brooks, Kristina; Kimerling, Jon. 1980. "GIS: a review of selected operational and functional capabilities." Department of Geography, Oregon State University, January.

Hoppe, Heinz Ulrich. 1990. "A Grammar-Based Approach to Unifying Task-Oriented and System-Oriented Interface Descriptions." *Mental Models and Human-Computer Interaction,* D. Ackermann and M. J. Tauber, editors. North-Holland: Elsevier Science Publishers BV, 353-373.

Jarke, Matthias and Vassiliou, Yannis. 1985. "A Framework for Choosing a Database Query Language." *Computing Surveys* 17, 3, 313-340.

Jarke, Matthias. 1986. "Current Trends in Database Query Processing." *On Knowledge Base Management Systems,* Michael L. Brodie and John Mylopoulos, editors. New York: Springer-Verlag, 111-119.

Kahn, Kenneth and Gorry, G. Anthony. 1977. "Mechanizing Temporal Knowledge." *Artificial Intelligence* 9, 87-108.

Katzeff, Cecilia. 1989. "Strategies for Testing Hypotheses in Database Query Writing." *Man-Computer Interaction Research,* F. Klix, N. A. Strietz, Y. Waern, and H. Wandke, editors. North-Holland: Elsevier Science Publishers B.V., 125-147.

Mackaness, William and Fisher, P. F. 1987. "Automatic Recognition and Resolution of Spatial Conflicts in Cartographic Symbolization." Proceedings of Auto-Carto 8, Baltimore, 709-718.

McGranaghan, M. 1989. "Context-Free Recursive-Descent Parsing of Location-Descriptive Text." *Proceedings of Auto-Carto 9,* Baltimore, April, 580-587.

Morse, Bruce W. 1987. "Expert System Interface to a Geographic Information System." *Proceedings of Auto-Carto 8,* Baltimore, March, 535-541.

Muller, J.-C., Johnson, R. D., and Vanzella, L. R. 1986. "A Knowledge-Based Approach for Developing Cartographic Expertise." Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, July, 557-571.

Nyerges, Timothy L. 1980. "Modeling the Structure of Cartographic Information for Query Processing." Unpublished PhD dissertation, Ohio State University, Columbus.

Nystuen, John D. "Identification of Some Fundamental Spatial Concepts." In *Spatial Analysis,* edited by Brian J. L. Berry and Duane F. Marble. Englewood Cliffs, NJ: Prentice-Hall Inc., 1968.

Overmyer, Ricky and Stonebraker, Michael. 1982. "Implementation of a Time Expert in a Database System." *SIGMOD Record* 12, 3, 51-59.

Robinson, G. and Jackson, M. 1985. "Expert Systems in Map Design." Proceedings of Auto-Carto 7, Baltimore, 430-439.

Schank, Roger C. and Rieger, Charles J. 1985. "Inference and the Computer Understanding of Natural Language." *Readings in Knowledge Representation,* Ronald J. Brachman and Hector J. Levesque, editors. Los Altos, California: Morgan Kaufmann Publishers, Inc., 119-140.

Plumb, Gergory A. 1988. "Displaying GIS Data Sets Using Cartographic Classification Techniques." Proceedings of GIS/LIS '88, San Antonio, December, Vol. 1, 340-349.

Quillian, M. Ross. 1985. "Word Concepts: a Theory and Simulation of some Basic Semantic Capabilities." *Readings in Knowledge Representation,* Ronald J. Brachman and Hector J. Levesque, editors. Los Altos, California: Morgan Kaufmann Publishers, Inc., 98-118.

Rucker, Rudy. 1987. *Mind Tools: the Five Levels of Mathematical Reality.* Boston: Houghton Mifflin Company.

Salmen, Larry J. 1978. "Natural Resource Analysis Techniques - Their Impact for DBMSs." *Harvard Papers on GIS,* Vol. 2, edited by G. Dutton. Reading, Massachusetts: Addison-Wesley.

Sinton, David. 1978. "The Inherent Structure of Information as a Constraint to Analysis: Mapped Thematic Data as a Case Study." *Harvard Papers on GIS,* Vol. 7, edited by G. Dutton. Reading, Massachusetts: Addison-Wesley.

Tomlin, Data. 1983. "Digital Cartographic Modeling Techniques in Environmental Planning." Unpublished PhD dissertation, Yale University.

Vasiliev, Irina. 1990. "Examples of the Treatment of Time as a Variable on Maps." Paper presented at the 1990 Conference of the Association of American Geographers.

Walshe, A. 1989. "Formal Methods of Database Language Design and Constraint Handlings." *Software Engineering Journal,* January, 15-24.

Webber, Bonnie Lynn. 1986a. "Natural Language Processing: A Survey." *On Knowledge Base Management Systems,* Michael L. Brodie and John Mylopoulos, editors. New York: Springer-Verlag, 353-364.

Webber, Bonnie Lynn. 1986b. "Questions, Answers, and Responses." *On Knowledge Base Management Systems,* Michael L. Brodie and John Mylopoulos, editors. New York: Springer-Verlag, 365-402.

Weibel, Robert and Buttenfield, Barbara P. 1988. "Map Design for Geographic Information Systems." Proceedings of GIS/LIS '88, San Antonio, December, Vol. 1, 350-359.

White, Marvin S. 1984. Technical Requirements and Standards for a Multipurpose Geographic Data System. *American Cartographer* 11, 1, 15-26.

Wu, Jian-Kang; Chen, Tao; and Yang, Li (1989). "A Versatile Query Language for a Knowledge-Based GIS." *International Journal of GIS* 3, 1, 51-57.

Qualitative Spatial Reasoning about Cardinal Directions[1]

Andrew U. Frank
National Center for Geographic Information and Analysis (NCGIA)
and
Department of Surveying Engineering
University of Maine
Orono, ME 04469 USA
FRANK@MECAN1.bitnet

Abstract

Spatial reasoning is very important for cartography and GISs. Most known methods translate a spatial problem to an analytical formulation to solve quantitatively. This paper shows a method for formal, qualitative reasoning about cardinal directions. The problem addressed is how to deduce the direction from A to C, given the direction from A to B and B to C. It first analyzes the properties formal cardinal direction system should have. It then constructs an algebra with the direction symbols (e.g., {N, E, S, W}) and a combination operation which connects two directions. Two examples for such algebras are given, one formalizing the well-known triangular concept of directions (here called cone-shaped directions) and a projection-based concept. It is shown that completing the algebra to form a group by introducing an identity element to represent the direction from a point to itself simplifies reasoning and increases power. The results of the deductions for the two systems agree, but the projection bases system produces more 'Euclidean exact' results, in a sense defined in the paper.

1. Introduction

Humans reason in various ways and in various situations about space and spatial properties. The most common examples are navigational tasks in which the problem is to find a route between a given starting point and an end point. Many other examples, such as decisions about the location of a resource, which translates in a mundane household question like "where should the phone be placed?", or the major problem of locating a nuclear waste facility require spatial reasoning. Military applications using spatial reasoning for terrain analysis, route selection in terrain, and so on. (Piazza and Pessaro 1990) are frequent. Indeed, spatial reasoning is so widespread and common that it is often not recognized as a special case of reasoning.

Spatial reasoning is a major requirement for a comprehensive GIS and several research efforts are currently addressing this need (Abler 1987, p. 306, NCGIA 1989, p. 125, Try and Benton 1988). It is important that a GIS can carry out spatial tasks, which include specific inferences based on

[1] Funding from NSF for the NCGIA under grant SES 88-10917, from Intergraph Corp. and Digital Equipment Corp. is gratefully acknowledged.

spatial properties, in a manner similar to a human expert and that there are capabilities that explain the conclusions to users in terms they can follow (Try and Benton 1988, p. 10). In current GIS systems, such spatial reasoning tasks are most often formalized by translating the situation to Euclidean geometry then using an analytical treatment for finding a solution. This is admittedly not an appropriate model for human reasoning (Kuipers 1978, p.143) and thus does not lead to acceptable explanations, but Euclidean geometry is a convenient and sometimes the only known model of space available for rigorous analytical approaches. A similar problem was found in physics, where the well known equations from the textbook were not usable to build expert systems. Using more qualitative than quantitative approaches, a formalization of the physical laws we use in our everyday lives was started, the so called 'naive physics' (Hayes 1985, Hobbs and Moore 1985, Weld and de Kleer 1990).

This paper addresses a small subset of spatial reasoning, namely qualitative reasoning with cardinal directions between point-like objects. We assume a 2-dimensional space and exclude radial reference frames, as is customary in Hawaii (Bier 1976). We want to establish rules for inference from a set of directional data about some points to conclude other directional relations between these. We follow McDermott and Davis (1984, p. 107) in assuming that such basic capabilities are necessary for solving the more complex spatial reasoning problems. A previous paper with the terms 'qualitative reasoning' in its title (Dutta 1990) is mostly based on analytical geometry. In contrast, our treatment is entirely qualitative and we use Euclidean geometry only as a source of intuition in Section 4 to determine the desirable properties of reasoning with cardinal directions.

Similarly, the important field of geographic reference frames in natural language (Mark, et al. 1987) has mostly been treated using an analytical geometry approach. Typically, spatial positions are expressed relative to positions of other objects. Examples occur in everyday speech in forms like "the church is west of the restaurant". In the past these descriptions were translated into Cartesian coordinate space and the mathematical formulations analyzed. A special problem is posed by the inherent uncertainties in these descriptions and the translation of uncertainty into an analytical format. McDermott and Davis (1984) introduced a method using 'fuzz' and in (Dutta 1988) and (Dutta 1990) fuzzy logic (Zadeh 1974) is used to combine such approximately metric data.

The problem addressed in this paper, described in practical terms, is the following: In an unknown country, one is informed that the inhabitants use 4 cardinal directions, by the names of 'al' 'bes' 'cel' and 'des', equally spaced around the compass. One also receives information of the type

> Town Alix is al of Beta, Celag is cel of Diton, Beta is des of Diton, Efag is cel of Beta, etc.

We show how one can assert that this is sufficient information to conclude that Alix is al of Efag.

Our concern is different from Peuquet (Peuquet and Zhan 1987), who gave 'an algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane'. She started with two descriptions of the shape of two objects given in coordinate space and determined the directional relationship (we say the cardinal direction) between the two objects. We are here concerned with several objects. Cardinal directions are given for some pairs of them and we are interested in the rules of inference that can be used to deduce others.

This paper lists a set of fundamental properties cardinal directions should have and defines what exact and approximate qualitative spatial reasoning means. It then gives two possible methods to construct a system of cardinal directions. They seem quite different, one based on a cone shaped or triangular area for a direction, the other based on projections, but they result in very similar conclusions. The projection based is slightly more powerful and easier to describe. The set of desirable properties are formally contradictory and contain some approximate rules, but these seem to pose more of a theoretical than a practical problem; however, clearly more research is necessary to clarify this point.

An approach that is entirely qualitative, and thus similar to the thrust in this paper, is the work on symbolic projections. It translates exact metric information (primarily about objects in pictures) in a qualitative form (Chang, et al. 1990, Chang, et al. 1987). The order in which objects appear, projected vertically and horizontally, is encoded in two strings, and spatial reasoning, especially spatial queries, are executed as fast substring searches (Chang, et al. 1988).

This work is part of a larger effort to understand how we describe and reason about space and spatial situations. Within the research initiative 2, 'Languages of Spatial Relations' of the NCGIA (NCGIA 1989) a need for multiple formal descriptions of spatial reasoning—both quantitative-analytical and qualitative—became evident (Frank 1990, Frank and Mark 1991, Mark and Frank 1990, Mark, et al. 1989). Terence Smith presented some simple examples during the specialist meeting .

> "The direction relation NORTH. From the transitive property of NORTH one can conclude that if A is NORTH of B and B is NORTH of C then A must be NORTH of C as well (Mark, et al. 1989)"

The organization of this paper is as follows: In Section 2 we introduce the concept of qualitative reasoning and relate it to spatial reasoning using analytical geometry; we define 'Euclidean exact' qualitative reasoning based on a homomorphism. In the following section, we list the properties of cardinal directions and in Sections 4 and 5 we discuss two different systems for reasoning with directions and compare them. We conclude the paper with some suggestions for future research.

2. Qualitative approach

2.1. Qualitative reasoning

In this paper, we present a set of qualitative deduction rules for a subset of spatial reasoning, namely reasoning with cardinal directions. In qualitative reasoning a situation is characterized by variables which 'can only take a small, predetermined number of values' (de Kleer and Brown 1985, p. 116) and the inference rules use these values and not numerical quantities approximating them. It is clear that the qualitative approach loses some information, but this may simplify reasoning. We assume that a set of propositions about the relative positions of objects in a plane is given and we have to deduce other spatial relationships (Dutta 1990, p. 351)

"Given: A set of objects (landmarks) and
A set of constraints on these objects.
To find: The induced spatial constraints".

The relations we are interested in are the directions, expressed as symbols representing the cardinal direction.

Without debating whether human reasoning follows the structure of propositional logic, we understand that there is some evidence that human thinking is at least partially symbolic and qualitative (Kosslyn 1980, Lakoff 1987, Pylyshyn 1981). Formal, qualitative spatial reasoning is crucial for the design of flexible methods to represent spatial knowledge in GIS and for constructing usable GIS expert systems (Buisson 1990, McDermott and Davis 1984). Spatial knowledge is currently seldom included in expert systems and is considered 'difficult' (Bobrow, et al. 1986, p.887).

In terms of the example given in the introduction, the following chain of reasoning deduces a direction from Alix to Efag:

1. Use 'Alix is al of Beta' and 'Efag is cel of Beta', two statements which establish a sequence of directions Alix - Beta - Efag.
2. Deduce 'Beta is al of Efag' from 'Efag is cel of Beta'
3. Use a concept of transitivity: 'Alix is al of Beta' and 'Beta is al of Efag' thus conclude 'Alix is al of Efag'.

We shall formalize such rules and make them available for inclusion in an expert system.

2.2. Advantage of qualitative reasoning

A qualitative approach uses less precise data and therefore yields less precise results than a quantitative one. This is highly desirable (Kuipers 1983, NCGIA 1989, p. 126), because

- precision is not always desirable, and
- precise, quantitative data is not always available.

Qualitative reasoning has the advantage that it can deal with imprecise data and need not translate it to a quantitative form. Verbal descriptions are typically not metrically precise, but are sufficient for finding the way to a friend's home, for example. Imprecise descriptions are necessary in query languages where one specifies some property that the requested data should have, for example a building about 3 miles from town. It is difficult to show this in a figure, because the figure is necessarily overly specify or very complex. Qualitative reasoning can also be used for query simplification to transform a query from the form in which it is posed to another, equivalent one that is easier to execute.

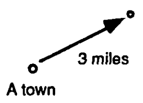


Figure 1: Overspecific visualization

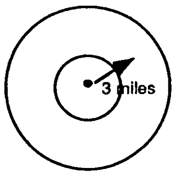


Figure 2: Complex visualization

In other cases, the available data is in qualitative form, most often text documents. For example, (Tobler and Wineberg 1971) tried to reconstruct spatial locations of historic places from scant descriptions in a few documents. Verbal information about locations of places can leave certain aspects imprecise and we should be able to simulate the way humans deduce information from such descriptions, (for example in order to automatically analyze descriptions of locations in natural science collections) (McGranaghan 1988, McGranaghan 1989, McGranaghan 1989).

2.3. Exact and approximate reasoning

We compare the result of a qualitative reasoning rule with the result we obtain by translating the data into analytical geometry and applying the equivalent functions to them. If the results are always the same, i.e., if we have a homomorphism, we call the qualitative rule **Euclidean exact.** If the qualitative rule produces results, at least for some data values, which are different from the ones obtained from analytical geometry, we call it **Euclidean approximate.**

$$\begin{tikzcd}
\begin{matrix}(P1, P2)\\(P2, P3)\end{matrix} \arrow[r, "dir"'] \arrow[d, "+"] & d1, d2 \arrow[d, "\infty"] \\
v + w \arrow[r, "dir"'] & d
\end{tikzcd}$$

Figure 3: Homomorphism

This is a general definition, which applies to the operation to combine two directions and deduce the direction of the resultant (introduced in 4.3, see figure 5). We establish a mapping from analytical geometry to symbolic directions using a function dir (P1, P2), which maps from a pair of points in Euclidean space to a symbolic direction (e.g., west). Vector addition, with the regular properties is carried to (i.e., replaced with) the symbolic combination ∞.

DEFINITION: a rule for qualitative reasoning on directions is called **Euclidean exact** (for short 'exact') if dir (P_1, P_2) is a homomorphism (Figure 3).

$$\text{dir}(P_1,P_2) \infty \text{dir}(P_2, P_3) = \text{dir}((P_1, P_2) + (P_2, P_3))$$

2.4. Formalism used

Our method is algebraic (specifically, a relation algebra) and the objects we operate on are the direction symbols S for south, E for west, not the points in the plane. Arguments involving pairs of points, standing for line segments between them, are used only to justify the desirable properties we list.

An algebra consists of

- a set of symbols D, called the domain of the algebra - comparable to the concept of data type in computer programming languages (e.g., D = {N, E, W, S}
- a set of operations over D, comparable to functions in a computer program (primarily operations to reverse and to combine directions), and
- a set of axioms that set forth the basic rules explaining what the operations do (Gill 1976, p. 94).

Specifically, we write (P_1, P_2) for the line segment from P_1 to P_2, and dir $(P_1,P_2) = d_1$ for the operation that determines the direction between two points P_1 and P_2, with d_1 the direction from P_1 to P_2 expressed as one of the cardinal direction symbols.

3. General properties of directions between points

We are interested in two types of operations applicable to direction:

- the reversing of the order of the points and thus the direction of the line segment (the inverse operation), and
- the combination of two directions between two pairs of consecutive points (the combination operation).

Using geometric figures and conclusions from manipulations of line segments, we deduce here properties of these two operations. These properties form then the basis for the qualitative reasoning systems defined in the next two sections.

We define direction as a function between two points in the plane that maps to a symbolic direction:

dir: p x p -> D.

The symbols available for describing the direction depend on the specific system of directions used, e.g., {N, E, S, W} or more extensive {N, NE, E, SE, S, SW, W, NW}.

In the literature, it is often assumed that the two points must not be the same, i.e., the direction from a point to itself is not defined. We introduce a special symbol, which means 'two points too close that a meaningful direction can be determined', and call it the identity element 0. This makes the function total (i.e., it has a result for all values of its arguments),

for all P dir (P, P) = 0.

3.1. Reversing direction

Cardinal directions depend on the order in which one travels from one point to the other. If a direction is given for a line segment between points

P_1 and P_2, we need to be able to deduce the direction from P_2 to P_1(Figure 4). Already (Peuquet and Zhan 1987) and (Freeman 1975) have stressed the importance of this operation: "Each direction is coupled with a semantic inverse". We call this 'inverse' (this name will be justified in 4.3.5) written as 'inv' .

inv: d -> d such that inv (dir (P_1, P_2)) = dir (P_2, P_1)

and

inv (inv (d)) = d because inv (inv (P1,P2)) = inv (P_2, P_1) = (P1,P2).

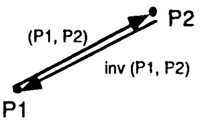


Figure 4: Inverse

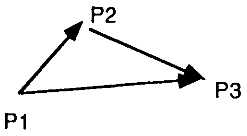


Figure 5: Combination

3.2. Combination

Two directions between two contiguous line segments can be combined into a single one. The combination operation is defined such that the end point of the first direction is the start point of the second.

comb : d x d -> d , always written in infix format: $d_1 \infty d_2 = d_3$

with the meaning:

dir (P_1,P_2) ∞ dir (P_2, P_3) = dir (P_1, P_3).

This operation is not commutative, but is associative, and has an identity and an inverse.

Combinations of more than two directions should be independent of the order in which they are combined (**associative** law) and we need not use parenthesis:

a ∞ (b ∞ c) = (a ∞ b) ∞ c = a ∞ b ∞ c (associative law)

This rule follows immediately from Figure 6 or from the definition of combination:

dir (P_1, P_2) ∞ (dir (P_2, P_3) ∞ dir (P_3,P_4)) =
dir (P_1, P_2) ∞ dir (P_2, P_4) = dir (P_1, P_4).
(dir (P_1, P_2) ∞ dir (P_2, P_3)) ∞ dir (P_3,P_4) =
dir (P_1, P_3) ∞ dir (P_3, P_4) = dir (P_1, P_4).

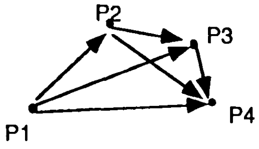


Figure 6: Associativity

The definition of an identity element states that adding the direction from a point to itself, dir (P_1,P_1) to any other direction should not change it.

$d \infty 0 = 0 \infty d = d$ for any d.

In algebra, an inverse to a binary operation is defined such that a value combined with its inverse, results in the identity value. From Figure 4 it follows that this is just the inverses of the given line segment:

dir $(P_1, P_2) \infty$ dir (P_2, P_1) = dir (P_1,P_1).

In case that two line segments are selected as in Figure 7, such that

dir $(P_1, P_2) = d_1$ and dir $(P_2, P_3) = d_2 =$ inv (d_1)

computing the combination

dir $(P_1, P_2) \infty$ dir $(P_2, P_3) = d_1 \infty$ inv $(d_1) = 0$

is an approximation and not Euclidean exact. The degree of error depends on the definition of 0 used and the difference in the size of the line segments - if they are the same, the inference rule is exact.

This represents a type of reasoning like New York is east of San Francisco, San Francisco is west of Philadelphia; thus the direction from New York to Philadelphia is 'too close' in this reference frame to determine a direction different from 'the same point' (which is defined here as an additional element of the possible values for a cardinal direction).

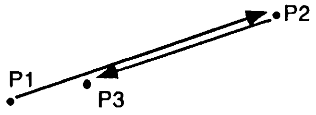


Figure 7: d ∞ inv (d)

We find that this combination is 'piece-wise' invertable:

inv (a ∞ b) = inv (a) ∞ inv (b).

Combinations of directions must have the special property that combining two line segments with the same direction results in the same direction. In a relation-oriented approach, this is a transitivity rule (as quoted in the introduction).

dir (P_1, P_2) = dir (P_2, P_3) = d then dir (P_1, P_3) = d
or short: $d \infty d = d$, for any d.

3.3. Summary of Properties of Cardinal Directions

The basic rules for cardinal directions and the operations of inverse and combination are:

- The combination operation is associative (1').
- The direction between a point and itself is a special symbol 0, called *identity* (1) (2')
- The direction between a point and another is the *inverse* of the direction between the other point and the first (2) (3').
- Combining two equal directions results in the same direction (*idempotent*, transitivity for direction relation) (3).
- The combination can be inverted (4).
- Combination is piece-wise invertible (5).

Properties of direction		Group properties	
$dir(P_1, P_1) = 0$	(1)	$d \infty (d \infty d) = (d \infty d) \infty d$	(1')
$dir(P_1, P_2) = inv(dir(P_2, P_1))$	(2)	$d \infty 0 = 0 \infty d = d$	(2')
$d \infty d = d$	(3)	$d \infty inv(d) = 0$	(3')
for any a, b in D exist unique x in D such that $a \infty x = b$ and $x \infty a = b$	(4)		
$inv(a \infty b) = inv(a) \infty inv(b)$	(5)		

Several of the properties of directions are similar to properties of algebraic groups or follow immediately from them. Unfortunately, the idempotent property (transitivity for direction relation) (3) is in contradiction with the remaining postulates, especially the definition of identity (3'). Searching for an inverse x for any $d \infty x = 0$, we find $x = d$ (using (3)) or $x = 0$ (using 3'), which contradicts the uniqueness of x (4). It is thus impossible to construct a system which fulfills all requirements at the same time. Human reasoning seems not to insist on associativity.

4. Cardinal directions as cones

The most often used, prototypical concept of cardinal directions is related to the angular direction between the observer's position and a destination point. This direction is rounded to the next established cardinal direction. The compass is usually divided into 4 major cardinal directions, often with subdivisions for a total of 8 or more directions. This results in cone shaped areas for which a symbolic direction is applicable. We limit the investigation here to the case of 4 and 8 directions. This model of cardinal direction has the property that 'the area of acceptance for any given direction increases with distance' (Peuquet and Zhan 1987, p. 66) (with additional references) and is sometimes called 'triangular'.

4.1. Definitions with 4 directional symbols

We define 4 cardinal directions as cones, such that for every line segment, exactly one direction from the set of North, East, South or West applies.

for every P_1, P_2 ($P_1 \neq P_2$) exist $d(P_1, P_2)$ with d in $D_4 = \{N, S, E, W\}$.

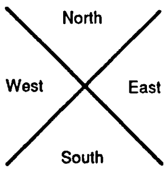


Figure 8: Cone-shaped directions

An obvious operation on these directions is a quarter-turn, anti-clock-wise (mathematically positive) q, such that

$q: d \rightarrow d$, with $q(N) = E, q(E) = S, q(S) = W, q(W) = N$

and four quarter turns are an identity:

$q\ (q\ (q\ (q\ (d))))= q^4\ (d) = d.$

Reversing a direction is equal to 2 quarter turns (or one half turn)

$\text{inv}\ (d) = q^2\ (d).$

Finally, we just define the combination of two directions, such that transitivity holds

$d \infty d = d$

but every other combination remains undefined.

These definitions would fulfill the requirements for the direction except that we did not define a symbol for identity. Very few combinations of symbols produce results.

4.2. Completion with identity

Introducing an identity element, we eliminate the restriction in the input values for the direction function

for every P_1, P_2 exist $d\ (P_1, P_2)$ with d in $D_5 = \{N, S, E, W, 0\}$.

A quarter turn on the identity element 0 is 0

$q\ (\ 0\ ) = 0$

and thus

$\text{inv}\ (\ 0\ ) = 0$	from $q(q(0)) = q\ (0) = 0$
$d \infty 0 = 0 \infty d = d$	from group properties
$0 \infty 0 = 0$	from $d \infty d = d$.

The inverse must further have the property that a direction combined with its inverse is 0

$d \infty \text{inv}\ (d) = 0.$

These definitions contain the previously listed ones as subset D_4 (not subgroup, because identity is not in the subset). Both the set D_5 and the subset D_4 is closed under the operations 'inverse' and 'combination'.

From the total of 25 different combinations, one can only infer 13 cases exact and 4 approximate; other combinations do not yield an inference result with these rules. Summarized in a table (lower case indicate approximate reasoning):

	N	E	S	W	0
N	N		o		N
E		E		o	E
S	o		S		S
W		o		W	W
0	N	E	S	W	0

4.3. Directions in 8 or more cones

One may use a set of 8 cardinal directions $D_9 = \{N, NE, E, SE, S, SW, W, NW, 0\}$, using exactly the same formulae. In lieu of a quarter turn, we define a turn of an eighth:

e (N) = NE, e (NE) = E, e (E) = SE, ... , e (NW) = N, e (0) = 0

with 8 eighth turns being the identity

e^8 (d) = d

and inverse now equal to 4 eighth turns

inv (d) = e^4 (d).

All the rules about combination of direction, etc., remain the same and one can also form a subset {N, NE, E, SE, S, SW, W, NW} without 0.

An approximate averaging rule combines two directions that are each one eighth off. For example, SW combined with SE should result in S, or N combined with E should result in NE.

e (d) ∞ -e (d) = d

with -e (d) = e^7 (d), or one eight turn in the other direction)

One could also assume that if two directions are combined that are just one eights turn apart, one selects one of the two (S combined with SE results in S, N combined with NW results in NW).

e(d) ∞ d = d and d ∞ e(d) = d

Human beings would probably round to the simple directions N, E, W, S, but formalizing is easier if preference is given to the direction which is second in the turning direction. This is another rule of approximate reasoning.

This rule can then be combined with other rules, for example to yield (approximate)

e(d) ∞ inv d = 0 and e(d) ∞ e (inv (d)) = 0.

In this system, from all the 81 pairs of values (64 for the subset without 0) combinations can be inferred, but most of them only approximately. Only 24 cases (8 for the subset) can be inferred exactly; 25 result in a value of 0 and another 32 give approximate results. We can write it as a table, where lower case denotes Euclidean approximate inferences:

	N	NE	E	SE	S	SW	W	NW	0
N	N	n	ne	o	o	0	nw	n	N
NE	n	NE	ne	e	o	o	o	*N*	NE
E	ne	ne	E	e	se	o	o	o	E
SE	o	e	e	SE	se	s	o	o	SE
S	o	o	se	se	S	s	sw	o	S
SW	o	o	o	s	s	SW	sw	w	SW
W	nw	o	o	o	sw	sw	W	w	W
NW	n	n	o	o	o	w	w	NW	N·W
0	N	NE	E	SE	S	SW	W	NW	0

5. Cardinal directions defined by projections

5.1. Directions in 4 half-planes

Four directions can be defined, such that they are pair-wise opposites and each pair divides the plane into two half-plains. The direction operation assigns for each pair of points a combination of two directions, e.g., South and East, for a total of 4 different directions. This is an alternative semantic for the cardinal direction, which can be related to Jackendoff's principles of centrality, necessity and typicality (Jackendoff 1983, p. 121). Peuquet pointed out that directions defined by half-planes are related to the necessary conditions, whereas the cone-shaped directions give the typical condition (Mark, et al. 1989, p. 24).

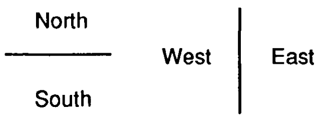


Figure 9: Two sets of half-planes

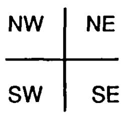


Figure 10: Directions defined by half-planes

Another justification for this type of reasoning is found in the structure geographic longitude and latitude imposes on the globe. Cone directions better represent the direction of 'going toward', whereas the 'half-plane' (or equivalent parts of the globe) better represents the relative position of points on the earth. However, the two coincide most of the time. To reach an object which is north$_{\textit{half-plane}}$ on the globe one has to go north$_{\textit{cone}}$.

For half-plane directions, one defines the cardinal directions as different from each other and E - W and N - S pair-wise inverse (Peuquet and Zhan 1987, p. 66). In this system, the two projections can be dealt with individually. Each of them has the exact same structure and we describe first one case separately and then show how it combines with the other.

The N-S case, considered the prototype for the two cases E-W and N-S has the following axioms:

for every P_1, P_2 ($P_1 \neq P_2$) dir_{ns} (P_1, P_2)= d_{ns} with d_{ns} in {N,S}

The inverse operation is defined such that inv (inv (d)) = d holds:

inv (N) = S, inv (S)= N.

Next we define the combination of two directions, such that transitivity holds:

for all d in {N,S} $d \infty d = d$ (which is $N \infty N = N$, $S \infty S = S$)

We now combine the two projections in N-S and E-W to form a single system, in which we have for each line segment one of 4 combinations of directions assigned.

D_4 = { NE, NW, SE, SW}

We label the projection operations by the directions they include (not the direction of the projection):

p_{ns}: d_4 -> d_{ns}, d_{ns} in {N, S}
p_{ew}: d_4-> d_{ew}, d_{ew} in {E, W}

and a composition operation

c: dns x dew -> d_9 such that c (p_{ns} (d), p_{ew} (d)) = d.

The rules for d_{ew} are the same as for d_{ns} explained above, replacing N by E and S by W:

$\text{inv}(E) = W, \text{inv}(W) = E$
$E \infty E = E, W \infty W = W.$

The inverse operation is defined as the inverse applied to each projection:

$\text{inv}(d) = c(\text{inv}(d_{ns}), \text{inv}(d_{ew}))$

and combination is similarly defined as combination of each projection

$d_1 \infty d_2 = c(d_{ns}(d_1) \infty d_{ns}(d_2), d_{ew}(d_1) \infty d_{ew}(d_2)).$

Unfortunately, combination is defined only for the four cases

$NE \infty NE = NE \qquad NW \infty NW = NW$
$SE \infty SE = SE \qquad SW \infty SW = SW$

and others, like

$NE \infty NW$

which should approximately result in N, cannot be computed. This system, lacking an identity, is not very powerful, as only 4 of the 16 combinations can be inferred.

5.2. Directions with neutral zone

We can define the directions such that points which are near to due north (or west, east, south) are not assigned a second direction, i.e., one does not decide if such a point is more east or west. This results in a division of the plane into 9 regions, a central neutral area, four regions where only one direction letter applies and 4 regions where two are used.We define for N-S three values for direction d_{ns} {N, P, S} and for the E - W direction the values d_{ew} {E, Q, W}.

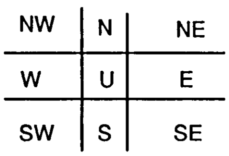


Figure 11: Directions with neutral zone

It is important to note, that there is no determination of the width of the 'neutral zone' made. Its size is effectively decided when the directional values are assigned and a decision is made that P_2 is north (not north-west or north-east) of P_1. We only assume that these decisions are consistently made. Similar arguments apply to the neutral zone of cone shaped directions, but they are not as important.

Allowing a neutral zone, either for the cone or projection based directions introduces an aspect of 'tolerance geometry'. Strictly, whenever we assign identity direction dir $(P_1, P_2) = 0$ for cases where $P_1 \neq P_2$ we violate the transitivity assumption of equality.

dir $(P_1, P_2) = 0$ and dir $(P_1, P_3) = 0$ need not imply dir $(P_2, P_3) = 0$

A tolerance space (Zeeman 1962) is mathematically defined as a set (in this case the points P) and a tolerance relation. The tolerance relation relates objects which are close, i.e., tol (A, B) can be read A is sufficiently close to B that we can or need not differentiate between them. A tolerance relation is similar to an equality, except that it admits small differences. It is reflexive and symmetric, but not transitive (as an equality would be)

tol (A, B)

tol (A, B) = tol (B, A).

A tolerance relation can be applied to geometric problems (Robert 1973).

Using the same methods as in 5.1 for the definition of the operations in each projection first and then combine them, we find for the inverse operation the following table:

d=	NE	N	NW	E	W	0	SE	S	SW
inv(d)=	SW	S	SE	W	E	0	NW	N	NE

The combination operation, again defined as the combination of each projection, allows one to compute values for each combination. Written as a table (again, lower case indicates approximate reasoning):

	N	NE	E	SE	S	SW	W	NW	0
N	N	NE	NE	e	o	w	NW	NW	N
NE	NE	NE	NE	e	e	o	n	n	NE
E	NE	NE	E	SE	SE	s	o	n	E
SE	e	e	SE	SE	SE	s	s	o	SE
S	o	e	SE	SE	S	SW	SW	w	S
SW	w	o	s	s	SW	SW	SW	w	SW
W	NW	n	o	s	SW	SW	W	NW	W
NW	NW	n	n	o	w	w	NW	NW	NW
0	N	NE	E	SE	S	SW	W	NW	0

The system is not associative, as

$(N \infty N) \infty S = N \infty S = 0$ but $N \infty (N \infty S) = N \infty N = N$.

In the half-plane based system of directions with a neutral zone, we can deduce a value for all input values for the combination operation (81 total), 56 cases are exact reasoning, not resulting in 0, 9 cases yield a value of 0, and another 16 cases are approximate.

6. Assessment

The power of the two systems which lack an identity element, the 4 direction cone-shaped and the 4 half-plane directional system, is very limited; most combinations cannot be resolved. The two systems with 8 direction and identity, the 8 direction cone-shaped and the 4 projection based directional system, are comparable. Each system uses 9 directional symbols, 8 cone directions plus identity on one hand, the Cartesian product of 3 values (2 directional symbols and 1 identity symbol) for each projection on the other hand. The reasoning process in the half-plane based system uses fewer rules, as each projection is handled separately with only two rules. The cone-shaped system uses two additional approximate rules which are then combined with the other ones. An actual implementation would probably use a table look-up for all combinations and this would not make a difference.

Both systems violate some of the desired properties. One can easily observe that associativity is not guaranteed, but the differences seem to not be very significant.

An implementation of these rules and comparison of the computed combinations with the exact value was done and confirms the theoretical results. Comparing all possible 10^6 combinations in a grid of 10 by 10 points (with a neutral zone of 3 for the projection based directions) shows that the results for the projection based directions are correct in 50% of the cases and in only 25% for cone-shaped directions. The result 0 is the outcome of 18% of all cases for the projection based, but 61% for the cone-shaped directions. The direction-based system with an extended neutral zone produces a result in 2% of all cases that is a quarter turn off, otherwise the deviation from the correct result is never more than one eighth of a turn (namely in 13% of all cases for cone-shaped and 26% for projection based direction systems). In summary, the projection based system of directions produces a result in 80% of all cases that is within 45° and otherwise the value 0.

7. Conclusions

This paper introduces a system for inference rules for completely symbolic, qualitative spatial reasoning with cardinal distances. We have first stressed the need for symbolic, qualitative reasoning for spatial problems. It is important to construct inference systems which do not rely on quantitative methods and need not translate the problem to analytical geometry, as most of the past work did. The systems investigated are capable of resolving any combination of directional inference using a few rules. Returning to our example in the introduction, we cannot only assert that Alix is al of Efag, but also that Alix is al-des from Diton and Celag, etc.

We used geometric intuition and the definition of a direction as linking two points. From this we deduced a number of desirable properties for a system to deal with cardinal directions. We use an algebraic approach and define two operations, namely inverse and combination. We found several properties, e.g.,

- the direction from a point to itself is a special value, meaning 'too close to determine a direction'
- every direction has an inverse, namely the direction from the end point to the start point of the line segment
- the combination of two line segments with the same direction result in a line segment with the same direction.

We defined the notion of 'Euclidean exact' and 'Euclidean approximate' as properties of a qualitative spatial reasoning system. A deduction rule is called 'Euclidean exact' if it produces the same results as Euclidean geometry operations would.

We then investigated two system for cardinal directions, both fulfilling the requirements for directions. One is based on cone-shaped (or triangular) directions, the other deals with directions in two orthogonal projections.

Both systems, if dealing with 4 cardinal directions, are very limited and when dealing with 8 directions, still weak. The introduction of the identity element simplifies the reasoning rules in both cases and increases the power for both cone and projection based directional systems. The deductions in this section use only the algebraic properties and does not rely on geometric intuition or properties of line segments.

Both systems yield results for all the 81 different inputs for the combination operation. But the projection based system more often yields an Euclidean exact result than the cone based one (49 vs. 25 cases). It also produces the value 0 less often (9 vs. 25 cases).

Another important result is that the two systems do not differ substantially in their conclusions, if definite conclusions can be drawn, i.e., not the value 0. This reduces the potential for testing with human subjects to find out which system they use, observing cases where the conclusion to use one or the other line of reasoning would yield different results.

We have implemented these deduction rules and compared the results obtained for all combinations in a regular grid. The projection based system results in 53% of all cases in exact results and in another 26% in results which are not more than 45° off. In 18% of all cases the application of the rules yields a value of 0. The results for the cone-shaped directions are less accurate. It will be interesting to see how this accuracy compares with human performance but also if it is sufficient for expert systems and for query and search optimization. The methods shown here can be used to quickly assess if the combination of two directions yields a value that falls within some limits and thus a more accurate and slower computation should be done.

There is not much previous work on qualitative spatial reasoning and several different directions for work remain open:

- Qualitative reasoning using distances,
- Combining reasoning with distances and directions,
- Hierarchical system for qualitative reasoning,
- Directions of extended objects, and
- Reasoning systems, human beings use.

Qualitative reasoning using distances - There is a good, mathematically based definition for distance measures expressed as real numbers. This can probably be carried over to qualitative distance expression, e.g., {Near, Far} or {Near, Intermediate and Far}, and rules for symbolic combinations similar to the one listed here deduced.

Combining reasoning with distances and directions - Combining the reasoning with directions and distances can be more than just combining two orthogonal systems; there are certainly interesting interactions between them (Hernández 1990). Most of the approximate reasoning rules are based on the assumption that the distances between the points discussed are about equal. This is not as unreasonable as it may sound, as directional reasoning is probably more often carried out regarding objects of the same import and thus at about the same distance.

Nevertheless, it is a weak assumption and further work should approach spatial reasoning on distances and then combine the two.

Hierarchical systems for qualitative reasoning - A system for reasoning with distances differentiating only two or three steps of farness is quite limited. Depending on the circumstances a distance appears far or near compared to others. One could thus construct a system of hierarchically nested neighborhoods, wherein all points are about equally spaced. Such a system can be formalized and may quite adequately explain some forms of human spatial reasoning.

Distances and directions of extended objects - The discussion in this paper dealt exclusively with point-like objects. This is a severe limitation and avoided the difficult problem of explaining distances between extended objects. Peuquet in (Peuquet and Zhan 1987) tried to find an algorithm that gives the same result than 'visual inspection'; however, visual inspection does not yield consistent results. It might be useful to see if sound rules, like the above developed ones, may be used to resolve some of the ambiguities.

What system of qualitative reasoning do humans use? - We can also ask, which one of the systems proposed humans use. For this, one has to see in which cases different systems produce different results and then test human subjects to see which one they employ. This may be difficult for the cone and projection based direction system, as their deduction results are very similar. Care must be applied to control for the area of application, as we suspect that different types of problems suggest different types of spatial reasoning.

Acknowledgements

Comments from Matt McGranagham and Max Egenhofer on a draft contributed considerably to improve the presentation and I appreciate their help. The thoughtful notes and suggestions from the reviewers are also greatly appreciated.

References

R. Abler. 1987. The National Science Foundation National Center for Geographic Information and Analysis. *International Journal of Geographical Information Systems* 1 (4) : 303-326.

J. A. Bier. 1976. *Map of O'ahu: The Gathering Place.* Honolulu, HI: University of Hawaii Press.

D. G. Bobrow, S. Mittal and M. J. Stefik. 1986. Expert Systems: Perils and Promise. *Communications of the ACM* 29 (9) : 880-894.

L. Buisson. 1990. "Reasoning on Space with Object-Centered Knowledge Representation". In *Design and Implementation of Large Spatial Databases.* Edited by A. Buchmann, O. Günther, T. R. Smith and Y.-F. Wang. 325 - 344. New York NY: Springer Verlag.

S.-K. Chang, E. Jungert and Y. Li. 1990. "The Design of Pictorial Databases Based Upon the Theory of Symbolic Projection". In *Design and Implementation of Large Spatial Databases.* Edited by A.

Buchmann, O. Günther, T. R. Smith and Y.-F. Wang. 303 - 324. New York NY: Springer Verlag.

S. K. Chang, Q. Y. Shi and C. W. Yan. 1987. Iconic Indexing by 2-D String. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (3) : 413 - 428.

S. K. Chang, C. W. Yan, T. Arndt and D. Dimitroff. 1988. An Intelligent Image Database System. *IEEE Transactions on Software Engineering* (May) : 681 - 688.

J. de Kleer and J. S. Brown. 1985. "A Qualitative Physics Based on Confluence". In *Formal Theories of the Commonsense World.* Edited by J. R. Hobbs and R. C. Moore. 109 - 184. Norwood NJ: Ablex Publishing Corp.

S. Dutta. 1988. "Approximate Spatial Reasoning". In *First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* 126 - 140. Tullahoma, Tennessee: ACM Press.

S. Dutta. 1990. "Qualitative Spatial Reasoning: A Semi-quantitative Approach Using Fuzzy Logic". In *Design and Implementation of Large Spatial Databases.* Edited by A. Buchmann, O. Günther, T. R. Smith and Y.-F. Wang. 345 - 364. New York NY: Springer Verlag.

A. U. Frank. 1990. "Spatial Concepts, Geometric Data Models and Data Structures". In *GIS Design Models and Functionality.* Edited by D. Maguire. Leicester, UK: Midlands Regional Research Laboratory, University of Leicester.

A. U. Frank and D. M. Mark. 1991. "Language Issues for Geographical Information Systems". In *Geographic Information Systems: Principles and Applications.* Edited by D. Maguire, D. Rhind and M. Goodchild. London: Longman Co. (in press).

J. Freeman. 1975. The modelling of spatial relations. *Computer Graphics and Image Processing* 4 : 156-171.

A. Gill. 1976. *Applied Algebra for the Computer Sciences.* Englewood Cliffs, NJ: Prentice-Hall.

P. J. Hayes. 1985. "The Second Naive Physics Manifesto". In *Formal Theories of the Commonsense World.* Edited by J. R. Hobbs and R. C. Moore. 1 - 36. Norwood NJ: Ablex Publishing Corp.

D. Hernández. 1990. *Relative Representation of Spatial Knowledge: The 2-D Case.* Report FKI-135-90. Munich FRG: Technische Universität München.

J. Hobbs and R. C. Moore ed. 1985. *Formal Theories of the Commonsense World.* Norwood HJ: Ablex Publishing Corp.

R. Jackendoff. 1983. *Semantics and Cognition.* Cambridge, Mass.: MIT Press.

S. M. Kosslyn. 1980. *Image and Mind.* Cambridge Mass.: Harvard University Press.

B. Kuipers. 1978. Modeling Spatial Knowledge. *Cognitive Science* 2 : 129 - 153.

B. Kuipers. 1983. "The Cognitive Map: Could it have been any other way?". In *Spatial Orientation.* Edited by H. L. Pick and L. P. Acredolo. 345 - 359. New York NY: Plenum Press.

G. Lakoff. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind.* Chicago, IL: University of Chicago Press.

D. M. Mark and A. U. Frank. 1990. *Experiential and Formal Representations of Geographic Space and Spatial Relations.* Technical Report (in press). : NCGIA.

D. M. Mark, A. U. Frank, M. J. Egenhofer, S. M. Freundschuh, M. McGranaghan and R. M. White. 1989. *Languages of Spatial Relations: Initiative Two Specialist Meeting Report.* Technical Report 89-2. : National Center for Geographic Information and Analysis.

D. M. Mark, S. Svorou and D. Zubin. 1987. "Spatial Terms and Spatial Concepts: Geographic, Cognitive, and Linguistic Perspectives". In *International Geographic Information Systems (IGIS) Symposium: The Research Agenda.* II-101 - II-111. Arlington, VA: NASA.

D. McDermott and E. Davis. 1984. Planning routes through uncertain territory. *Artificial Intelligence* 22 : 107-156.

M. McGranaghan. 1988. "Prototyping an Herbarium Mapping System". In *ACSM-ASPRS Annual Convention.* 232 -238. St. Louis: ACSM.

M. McGranaghan. 1989. "Context-Free Recursive-Descent Parsing of Location-Descriptive Text". In *Auto-Carto 9.* 580 - 587. Baltimore MD: ACSM.

M. McGranaghan. 1989. "Incorporating Bio-Localities in a GIS". In *GIS/LIS '89.* Orlando FL: ACSM.

NCGIA. 1989. The U.S. National Center for Geographic Information and Analysis: An overview of the agenda for research and education. *International Journal of Geographic Information and Analysis* 2 (3) : 117-136.

D. Peuquet and C.-X. Zhan. 1987. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in a plane. *Pattern Recognition* 20 : 65-74.

P. A. Piazza and F. Pessaro. 1990. "A Cognitive Model for a "Smart" Geographic Information System". In *GIS World Annual Source Book.* GIS World.

Z. Pylyshyn. 1981. The Imagery Debate: Analogue Media versus Tacit Knowledge. *Psychological Review* 87 : 16-45.

F. S. Robert. 1973. Tolerance Geometry. *Notre Dame Journal of Formal Logic* 14 (1) : 68-76.

W. R. Tobler and S. Wineberg. 1971. A Cappadocian Speculation. *Nature* 231 (May 7) : 39-42.

P. D. Try and J. R. Benton. 1988. *Critical Investment Strategy Guidance for Artificial Intelligence Development in support of the Realistic Battlefield (Terrain/Weather/Obscurants) - A report of the Workshop on Artificial Intelligence Research for Exploitation of the Battlefield Environment.* STC Technical Report 3087. Hampton VA: Science and Technology Corp.

D. S. Weld and J. de Kleer ed. 1990. *Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

L. A. Zadeh. 1974. "Fuzzy Logic and Its Application to Approximate Reasoning". In *Information Processing.* : North-Holland Publishing Company.

E. C. Zeeman. 1962. "The Topology of the Brain and Visual Perception". In *The Topology of 3-Manifolds.* Edited by M. K. Fort. 240 -256. Englewood Cliffs, NJ: Prentice Hall.

Improving Spatial Analysis in GIS Environments

Geoffrey Dutton
Spatial Effects
150 Irving Street
Watertown MA 02172 USA
qtm@cup.Portal.Com

Abstract

Current GIS technology tends to impede problem-solving for many of its users and is difficult for vendors to develop and support. Why this may be the case and what might be done about it is explored in this paper. Three problems shared by most commercial GIS' are identified and examined: inadequate data models; inferior application development tools; insufficient on-line expertise. It is argued that each factor inhibits robust spatial analysis, and limits the usability of analytic and cartographic GIS outputs. Suggestions based on recent research directions and emerging software engineering practices are given for addressing deficiencies in these three realms. Certain properties of and synergies among data models, application toolkits and expert systems are explored as keys to improving data and knowledge management, user interaction and spatial analysis in geographic information systems.

Introduction

Geographic Information Systems (GIS) are a protean emerging technology involving many primary data sources (spatially sampled measurements of the natural and human environment, surveying and photogrammetric data, digitized maps and remotely-sensed images), diverse data structures (points, polygons, networks, rasters, quad - or whatever - trees), complex databases (geometric, topological, attribute and metadata, relational, hierarchical, distributed and hypermedia), evolving analytic methods (network and surface synthesis and analysis, feature extraction, spatial overlay, temporal change and other attribute analysis), high-quality cartography (2+D rendering, what-if graphics, engineering plans, thematic maps) and high sensitivity to data quality (positional accuracy, feature coding, resolution and scale effects, and spatial/temporal aliasing). GIS emerged from laboratory gestation in the mid-1980's to confront an explosion of environmental challenges and applications. But while computing hardware capable of manipulating complex spatial data is increasingly within the reach of users, GIS developers do not completely agree about how spatial software should best be built, spatial data structured, applications wrought and spatial analysis conducted. This is in part due to a lack of high-level tools, but also is a consequence of relying upon low-level constructs that are proving increasingly inadequate. Twenty-five years of GIS progress should not prevent us from re-evaluating our basic assumptions and prevailing models. Doing this might broaden our perspectives and invigorate our technology.

Solving spatial problems

GIS evolved from early attempts to conduct spatial analysis using digital cartographic data in vector and raster form. Success has always been limited by the amount of information encoded into cartographic databases. Sets of points, lines and polygons, while fully defining entities in a "geographic matrix" (Berry, 1964), fail to model spatial relationships among them. To such descriptions topology has been added (Corbett, 1977), defining how cartographic entities connect to one another. Other, potentially valuable capabilities not now in use have been proposed: data quality documentation (Chrisman, 1984); global hierarchical spatial indexing (Dutton, 1989; Goodchild and Yang, 1989; Fekete, 1990); temporal data management (Langran, 1989). While these approaches could be incorporated into existing systems, the effort and cost required would be formidable; a new generation of software may be needed instead.

In any case, alternatives to current spatial data models are already needed. In the author's view, if academia and industry are to meet the challenge of supporting global environmental science, developers will have to retool GIS databases at a rather basic level. This is because so many of the "coverages", "partitions" and "projects" by which GIS's administrate databases are modeled as planar, cartesian chunks of the world, represented as maps. Although many systems can perform transformations between projections and into latitude and longitude, this is usually only done to "register" coverages by mapping coordinates into a preferred planar projection. Certain special-purpose and in-house GIS's store spherical coordinates in their databases, and thus are in principle capable of working on a global scale. But the mathematics involved in manipulating such data can be costly, and the ambiguities inherent in attempting to positively identify points and their loci will continue to confound applications, especially when data quality information is lacking or goes unused.

Despite the fact that most vendors heavily promote "solutions" (sets of niche applications) GIS isn't really the sum of vertical markets; it's a technical infrastructure (like DBMS) upon which applications may be erected. While different applications of spatial data have unique if not conflicting analytical requirements (what methods do crop assessment and network analysis have in common, for example), their implementation usually insures that their data will remain incompatible. It may be that geographic data deserves support at the system level, commensurate to the facilities hardware vendors now provide for manipulating text, numbers, tables, images, abstract datatypes and user interfaces. Given sufficiently capable data structures, methodologies, and advice packaged in forms accessible to end-users, more robust and specialized GIS applications could be generated more easily; these would effectively combine known spatial analytic and cartographic methods with canonical ways of storing, retrieving and manipulating spatial data, guided by facts about data domains and rules that apply to them.

Solutions beget problems

In a recent paper describing a systems-level architecture for supporting use of hypertext within and across diverse applications, Kacmar (1989) identifies several problems that existing hypertext implementations exacerbate:

> Current hypertext systems have attempted to a provide an all-inclusive work environment for the user. However, few systems have been able to realize this goal. Thus, users are required to utilize several applications for their activities and must enter and exit applications in order to accomplish specific tasks. The hypertext system becomes yet another application and other user interface mechanism which must be learned and used. (Kacmar, 1989, p. 98)

One can substitute "GIS" for "hypertext system" in this text without changing its sense, just its context. In contrast to hypertext authors, however, GIS users must be more than casually aware of the nature of the database and data structures their systems manipulate, due to the various special properties of spatial entities that they model (hypertext data mainly consists of text fields linked as a semantic net having abstract, user-imposed and self-specified "spatial" relationships). That is, while all the semantics of a hypertext database are user-specified, much of the semantics of spatial information is given or constrained by physical laws, common law, administrative regulations, data structures and measurement theory. When a GIS is used to model spatial semantics, the ways in which rules are applied and information is communicated to users tend to vary greatly in completeness, consistency and complexity.

When enhancing their systems, GIS vendors tend to maintain compatibility with earlier versions (to safeguard users' investments), even though it might be technically advisable to radically redesign applications. One result is that new commands or modules tend to be added on top of or alongside of existing ones; this can steepen the learning curve for affected applications, and still not assure that the tools provided will serve users' purposes. Only highly-motivated users may exercise the more complex applications and options, often to discover that useful features or parameters in enhanced applications aren't available in other, related contexts. As GIS data grows more complete and complex (that is, as systems incorporate more information about spatial semantics), the "span of control" confronting users will also increase, and vendors will have to work hard to make systems uniformly and consistently usable. This will be necessary regardless of what type of user interface is involved (command lines, menus, direct manipulation, hypermedia).

Problems (sort of) fade away

Like hypertext, GIS is not yet a mature technology. This should not, however, be used as an excuse for perpetuating difficulties involved in learning and applying GIS. Cooke (1989) argues that we are about to enter the "post-GIS era", in which the bulk of data capture activities will have already been accomplished or become relatively automated. In such a milieu, our attention will naturally turn toward modeling and analyzing spatial phenomena; much of the output from GIS will be non-graphic (such as inventories of property, estimations of resource acquisition and operating costs, environmental status of specified areas, or the address of the nearest elementary school), and users will demand error reports, confidence limits and sensitivity analyses for data they analyze. They will also need assistance in browsing through ever-larger spatial inventories, in formulating queries to extract data relevant to their purposes and in specifying the steps necessary to perform particular analyses. Still, progress will inevitably occur, in Cooke's view, leading to a flourishing of environmental applications:

Technical issues of digitizing, coordinate conversion, map-edge matching and topological editing will fade into history. We finally will be able to turn our creative energies to solving real problems, of which we have plenty. Pollution, ozone depletion, the greenhouse effect, all are exacerbated by the inefficient logistical operations resulting in unnecessarily burning fossil fuel. (Cooke, 1989, p. 55)

But will data maintenance tasks and data quality problems ever fade away, and if so, what will cause this to happen? Major users of GIS and CADD, such as local governments, seem unable to keep up with the pace of change in their jurisdictions, and it is hard to imagine that their digitizing activity will ever cease. New versions of TIGER and other base files will be issued, but it may never become trivial to integrate them with an organization's database. It seems apparent that spatial data is not going to get easier to handle just because it is growing more complete and accurate, and can be obtained in digital form, even though much drudgery may be eliminated for users. Integrating spatial data isn't inherently difficult; it has been made difficult by a plethora of local coordinate systems, differing (and inadequate) data models and data structures, primitive data interchange standards, insufficient data quality information and a disinclination to use it. Metaphorically speaking, we have built a maze of datatypes, tools, techniques and topology, and are getting frustrated because we can't find our way out. Perhaps a good sales slogan for our industry would be "Lose yourself in GIS (it's easy)!"

Retooling GIS

To save GIS from crippling itself and to help users meet the challenges that their work presents, GIS researchers and developers need to take a critical look at the factors that limit the effectiveness, reliability and usability of current technology. In the author's view, such reexaminations should focus on three problem areas, which seem to map to three scales of software engineering and involve three groups of actors. These key problem areas, or realms, are:

1. Data models;	How spatial data is represented for computation
2. Application toolkits;	Better ways of constructing custom software
3. On-line expertise.	Access to knowledge about data and methodology

Table 1 describes an operational context for these realms.

Table 1

Three Critical Realms of GIS Technology and Application

REALM	SCALE	ACTORS	CURRENT PROBLEMS
Data models	Micro	Researchers	Need to improve, standardize
Application toolkits	Meso	Developers	More flexible, customizable
On-line expertise	Macro	Users et al	Better help, reports, advice

"Scale" refers to the extent of software modules that reify the realms (e.g., functions, libraries, subsystems). "Actors" are those professionals who are most central to implementing the realms. As many GIS professionals have played all the above roles at one time or another, they should be aware of the impact of each of the realms on actors' activities. This is not to exclude the effects of related topics (such as object-oriented software, numerical algorithms, hypermedia and user interface design) on the state of the art. But these and other realms are receiving attention not only in the GIS community but in software engineering in general. The three realms listed in Table 1 pose direct challenges to GIS, and will have to be addressed by both our industrial and research enterprises.

Data Models

Most GIS employ data models (the conceptual organization used for spatial and aspatial data) inherited from computer cartography, CADD and civil engineering, image processing and management information systems. They tend to be, as Cooke (1989) points out, based on the "map-as-graphic" paradigm rather than "map-as-database". The variety of data models (e.g., image, object, network, layer) and many variations in the data structures used to implement them (arc-node, grids, quadtrees, TINS) has led to difficulties in comparing and exchanging datasets, even when "standard" interchange formats are used; some relationships may not be encodable, hence are lost, unless they are later reconstructed — often at great expense. It has also bred a somewhat cavalier attitude that encourages incompatible data structures to coexist, even within the same database. Thus, the decades-old "vector v. raster" debate, having never been resolved, has been made moot by concessions that each structure has unique strengths; as one may always be converted to the other, there is no need to make a choice other than for tactical expedience.

While vector- and raster-encoded spatial data may in some sense be equally good (both are certainly useful), they can also be considered to be equally bad. Raster files tend to be bulky, unstructured and insensitive to variations in data density. Vector data structures can be cryptic and complex to manipulate, and often fail to express variations in data quality. Both models tend to ignore or filter out important aspects of spatial structure in abstracting geographic data. To compensate for these losses, ingenious and sophisticated methods have been built into GIS's: "fuzzy overlay", "rubber-sheeting", and resampling and filtering in both the spatial and spectral domains. But rarely are the tribulations that these methods are designed to overcome traced back to their source: loss of information in exchanging, digitizing and scanning maps and images due to impoverished data models.

Remote sensing technology has succeeded in recovering remarkably useful data from rather imperfect platforms, and has developed a canon of tools and techniques for correcting geometric and radiometric errors and finding structure in image data, often implemented as black-box functions which are tricky to integrate and easy to misuse.[1] Cartographic digitizing methods have also improved, especially in terms of avoiding, identifying or compensating for operator blunders and errors (White and Corson-Rikert, 1987). Most GIS's still express the quality of digitized data rather simplistically, relying on a few global parameters (such as U.S. map accuracy standards express), which fail to express local variations in spatial uncertainty inherent to the phenomena being captured. Even if this information were to be provided, prevailing GIS data models tend to have no place to put it, and their analytic procedures generally make little use of data quality information in their deliberations.

This state of affairs represents an ironic twist of autocarto evangelism[2]: After SYMAP and other software enabled digital thematic mapping, a lot of effort went into explaining to cartographers what polygons were, and this eventually instilled in them an abiding attachment to coordinates, which for awhile they resisted, then embraced just around the time they were told that polygons weren't enough, and they needed to learn about topology. After DIME (A.D.) embedded map networks in a rigorous mathematical framework (Cooke and Maxfield, 1967), it seemed for awhile that topology would solve most spatial data-handling problems, because coverages could now be verified to be complete and correct. The naîvete of this presumption was made evident by the advent of GIS; as soon as analysts began to merge and overlay

[1] Beard (1989) identifies *use error* as the "neglected error component" in GIS applications. While human error is difficult to quantify, it seems apparent that the more commands, options and parameters that a user confronts at a given moment, the greater the likelihood that (s)he will make a mistake.

[2] Many highlights and sidelights of the development of computer cartography and GIS are related in a recent issue of *The American Cartographer* (vol. 15, no. 3, July 1988), subtitled "Reflections on the revolution: the transition from analogue to digital representations of space, 1958-1988".

map data derived from different sources they found that while their computer could connect complex mazes of dots and lines into a single network and name all the objects therein, many if not most of these often turn out to be artifacts that have no basis in reality.

Today, c. 24 A.D., the source of such hassles is widely acknowledged to stem from failures to maintain data quality information within spatial databases. What is *not* as widely appreciated is that this may directly issue from twenty-five years of representing spatial locations as two- and three-dimensional coordinate tuples that have no inherent scale, only precision. This peculiar myopia has been termed *the fallacy of coordinates* (Dutton, 1989b), the (often unconscious) leap of faith that coordinates actually exist. It describes, but fails to explain, how entire professions and much software have come to accept point coordinates as if they were natural phenomena, like pebbles or protons. Still, it is not surprising that coordinates are reified in a culture which regards land as real estate, in which inches of frontage can cost dearly and where boundaries need not hew to visible landmarks. It is odd and rather distressing to have to preach the heresy that coordinates are the antichrist of spatial data handling, given that computer cartography and GIS have been around for a quarter of a century. But there are times in any walk of life when conventional wisdom bears reexamination.

Part of the reason why coordinates prevail is due to the view that digital spatial data represents maps, which in turn represent the world. It seldom seems to occur to GIS developers that they might better serve users by regarding maps as products of, rather than as the basis for their systems. As a result, most GIS's use "cartographic data structures" (Peucker and Chrisman, 1975), which are good at encoding features on maps, but which eventually fail to represent much of the evidence available about distributions of things and events on our planet. Failure to handle temporality is one resultant problem (Langran, 1989), but there are others. Goodchild (1988) offers an example of how technical factors have shaped and constrained the development of computer cartography and GIS:

> In the case of forest inventory maps, the need for accurate inventory clearly overrides any question of cartographic clarity and ease of perception. However forest inventories continue to be mapped using bounded areas to portray homogeneous forest stands, suggesting in this case that technological constraints, specifically the inability to show transition or heterogeneity, have outweighed any more abstract cartographic principles. ... The consequences of those constraints can be rationalized as intelligent choices, and are so fundamental that it is difficult to consider alternatives, but they are in actuality severely restricting, and influence not only the way we portray the world but also the way we observe it. (Goodchild, 1988, ps. 312 & 317)

While acknowledging that any data model has limitations, Goodchild points to emerging alternative spatial data structures which might overcome problems inherent in electronic emulations of pen-and-paper technology:

> In one sense, [quadtrees] represent a departure from fixed scale in the form of fixed pixel size in rasters or fixed levels of spatial generalization in vectors. They are non-intuitive in that they correspond to no conventional pictorial view, but have meaning only as digital representations. In quadtree data structures we are beginning to see the emergence of a genuinely new technology in which methods have no obvious conventional analogues. At the same time the constraints imposed by the technology are radically different. (Goodchild, 1988, p. 316)

It is worth stressing again that limitations of map-as-graphic and point-line-area paradigms cannot be overcome without purging ourselves of the notion that locations in the real world are dimensionless points; neither will we make real progress by continuing to pretend (at least in our databases) that the Earth is flat and that we occupy its lower left-hand corner. The former prejudice prevents us from modeling spatial distributions in ways that capture their indeterminate and scale-dependent qualities. The latter assumption inhibits development of GIS databases and techniques that can deal with information from diverse sources and operate at continental or global scales. Such databases are being built (often haphazardly) at accelerating rates, and global GIS issues can no longer be swept under cartesian rugs. According to Tomlinson:

> The ability to integrate data with a variety of formats (raster, vector, street address and tabular) from different sources, at different levels of reliability, at different scales, by people with different skills, using different computers, in different countries, connected by communication networks, is a very real requirement in the foreseeable future. (Tomlinson, 1988, p. 259).

One recent approach to meeting these challenges is to design spatial databases that represent locations hierarchically, by indexing to positions occupied by vertices and faces of nested polyhedra, successively approximating the surface of a planet. While polyhedral map projections are not new (going back at least to the time of the artist Albrecht Dürer), the idea of tessellated, polyhedral data storage hierarchies is probably less than a decade old and still relatively unexplored (Peuquet, 1988); few schemes specific to GIS have been proposed (Dutton, 1984[3], 1989a[4]; Goodchild and Yang, 1989[5]; Fekete, 1990), and none of these have been demonstrated in fully operational contexts.

While not nearly enough research as been undertaken to verify expectations, one can anticipate a number of benefits that might flow from implementing such data models. Casting coordinates into hierarchical planetary tessellations might mitigate many of the data-handling problems that plague current GIS technology and short-circuit spatial analysis: using such methods, any planetary location can be canonically encoded, regardless of where it is or the resolution at which it is identified; features could be modeled more readily, matched and integrated more easily, with certifiable accuracy; different datasets encoding diverse locations could be merged with greater confidence and ease. Multi-resolution storage lets primitive data elements specify their inherent accuracy; collections of such elements can model complex objects, which can be retrieved at a variety of appropriate scales. Such collections may be cast into both raster and vector formats, but will have other

[3] The *geodesic elevation model (GEM)* is a dual (cube-octahedron) polyhedral tessellation designed to encode terrain relief of planets using two alternating ternary hierarchies. This horizontal organization was coupled with difference-enoding of elevations to provide a compact, self-calibrating and scale-sensitive representation of topographic relief. *GEM* was simulated, but never implemented.

[4] The *Quaternary Triangular Mesh (QTM)* scheme derives from *GEM*. *QTM* is a region quadtree composed of triangles. It represents a planet as an octahedron comprised of 8 quaternary triangular grids, and can encode locational data both as hierarchies and sequences. Collections of such codes can be structured to represent geographic objects at specific scales. *QTM* has fractal properties, which may be exploited by modeling locations as basins of attraction (*attractors*), hexagonal regions centered on *QTM* grid nodes and composed of six adjacent triangular tiles, to which all locations in their domain alias at some level of detail. Attractors knit together adjacent *QTM* domains, and may aid in identifying and preventing "slivers" when overlaying vector-encoded map features.

[5] Goodchild and Yang's *Triangular Hierarchical Data Structure (THDS)* is a variation of *QTM*. What distinguishes *THDS* from *QTM* is the simplicity of (a) its facet numbering scheme (which ignores attractors), (b) its geodesic computations (all subdivision occurs on planar octahedron facets), and (c) its indexing algorithms (although transformations between the two orderings have been developed).

qualities (deriving from the properties of recursive geodesic tessellations) that may be exploited by new species of data structures and algorithms.

Application Toolkits

Spatial analysis tasks range from primitive computations such as nearest-neighbor identification and interpolation of point sets to multi-stage simulations of urban growth and multi-layer land suitability studies. Analysts often engage in *ad hoc* explorations of data before (or in lieu of) settling down to a standard methodology. Many vendors hasten to fulfill requests from users for new analytic capabilities, resulting in ever-greater arrays of GIS applications, commands and options. Some systems let end users cobble together the functionality they need as procedures coded in a macro language provided by the vendor, similar to the way most spreadsheet applications are built. Larger vendors have application consultants who extend existing applications by writing source code modules and linking them in as new commands. Such customization activity is commonplace for any number of reasons, including:

- GIS, like CADD systems, must operate in many diverse environments;
- Many spatial analysis procedures are sensitive to details of data models;
- Data items (e.g., "attributes") may reside in various foreign databases;
- GIS applications and analytic methodologies are still evolving.

As GIS's and their applications proliferate, the pace and economics of software development make it difficult for vendors to keep up with analysts' demands for new functionality, and new users face steeper learning curves, requiring increasing amounts of documentation, training and time in order to become productive. In such a milieu, costs of developing and enhancing applications tend to increase, regardless of who performs the work. These seem to be problems generic to all applications software systems, and are by no means unique to GIS. They do, nevertheless, represent real impediments to improving GIS analytic capabilities.

Segments of the software industry have begun to adopt a variety of computer aided software engineering (CASE) tools to manage various stages of the software lifecycle, from requirements analysis and functional specification to user interface construction and code generation. In addition, end-user environments for building microcomputer applications are gaining in popularity, particularly those that provide direct-manipulation visual interactive programming (VIP) capabilities (Sabella and Carlbom, 1989). Many such tools are based on object-oriented (OOP), rather than procedural programming styles (Cox, 1987). Proponents of this approach maintain that it reduces the need for debugging, clarifies program structure, speeds development and increases software reusability. Even if these claims are true, OOP methodology may not help developers create better algorithms, not be as computationally efficient as traditional procedural code, and may prove hard to interface with the miscellany of data structures typically found in GIS environments.

The jury is still out concerning the utility of CASE, VIP and OOP to software developers and end-users. But it is apparent that many of the tasks traditionally assumed by developers and vendors are, by economic necessity and by popular demand, shifting to user domains. It also seems apparent that spatial analysis in a GIS environment is most effectively conducted when people most familiar with the problem domain actively participate in software development. This points to the provisional conclusion that GIS vendors ought to be encouraged to provide visual programming environments in which users can craft analytic software and other finicky functions, such as database interfaces, data filters and thematic maps. Such modules should be made as easy as possible to create, and this implies that objects in GIS databases need to represent themselves with appropriate data structures and manipulate themselves with appropriate methods, in transparent ways.

An informative account of modeling and visualizing geological reservoir data (Sabella and Carlbom, 1989) describes a set of tools called *Gresmod,* a laboratory testbed for rendering heterogeneous collections of geometric objects. In this prototype, users at a Xerox workstation worked with *Gresmod*'s OOP modeling environment and diagrammatic interface to specify a data base of geometric structures (curves, planes, and solids encoded as octrees, representing oil reservoirs), perform limited analysis, and render it in 3D on a high-performance graphic display. *Gresmod* was written in an OOP language called *Strobe,* which builds a set of knowledge bases for class hierarchies, attributes and methods. *Gresmod*'s direct-manipulation user interface (also knowledge-based), was built with the *Grow* toolkit for customizing the *Impulse-88* UIMS used by Sabella and Carlbom. *Strobe,* in turn, is implemented in Interlisp-D; it includes a graphic editor for objects, a run-time environment, and manages knowledge bases. The study provides an "object lesson" in how diverse forms of spatial data can be manipulated in unified fashion in a visual interactive environment. It shows how spatial analysis can be enhanced when tools for data modeling, application building and knowledge management are provided.

Commercial GIS technology can't do this yet, but is headed in these directions. Providing such environments will propel fundamental conceptual and architectural changes in system design. It is both technically possible and commercially expedient to transfer responsibility for application development to user communities, rather than continuing to place this staggering burden on vendors. This means that vendors must move away from offering *solutions* and toward providing *toolkits* that enable users to solve their particular, idiosyncratic problems. This approach is being adopted by vendors of high-performance workstations for data visualization (Upson *et al*, 1989); users may create or modify visual models by editing on-screen dataflow diagrams and property sheets, without the need for much if any program coding.

On-line Expertise

Printed and on-line documentation for most applications software packages generally include tutorials on how to get started, detailed command descriptions and one or more examples of data processing using sample datasets. As a GIS may include dozens of functions (ranging from map digitizing and topological editing, through feature and attribute definition, analytic procedures and display formatting, not forgetting project management activities), each of which may have dozens of commands, and its printed documentation can easily fill a four-foot shelf. While the general workflow at most GIS installations has many common and predictable aspects (digitize, edit, structure, extract, merge, overlay, analyze, report, map), the details of system configurations and users' data and applications differ enough to limit the utility of vendors' printed documentation and on-line help.

While users can normally refer to manuals to find information that answers certain types of questions ("What does command X do?"; "What commands do *Y* to data of type *Z*?"), they often can't find answers to many other kinds of questions ("What do I do next?"; "How will executing command X affect my database?"; "How much error might attribute *I* of object *J* have?"). The number of paths users can follow through a complex application is essentially unlimited. Certain paths may yield equivalent results with some data, but produce quite different results with other data; such discrepancies may derive from the content and quality of data being processed, parameters and options selected by users, the order in which commands are invoked, software bugs or be totally inexplicable and unreplicatable (known to hackers as the POM — Phase of the Moon — effect). And while the underlying database machinery may be able to roll back transactions that go awry, the amount of processing and operator time wasted in doing this can be a great drain on users (GIS transactions are typically quite lengthy, and all work performed between a check-out and a roll-back may have to be sacrificed, even if most of it produced correct results). While such unpleasant surprises may be the results of encountering software bugs or pathological data, most stem from the impossibility of grasping all the *normal* consequences of executing particular commands in particular sequences on particular sets of data.

The sheer complexity of working in a GIS environment is likely to increase as applications become more analysis-intensive. Printed and on-line documentation cannot even in principle address this problem, because most of the complexity is combinatorial, most of the questions context-sensitive. And while there are people who are skilled at dealing with such matters, such knowledge takes a long time to acquire and tends to be idiosyncratic and difficult to transfer to others. Even though the population of GIS gurus is growing, expertise of this type will remain a scarce resource for both vendors and users for the foreseeable future.

Many data-handling decisions are made automatically by software, based on options and parameters input by users (or their default values) and variables and constants stored in datasets (or their estimates). These are applied either on a case-by-case basis in the course of processing data items, or less frequently, to select a processing strategy for performing a given task (rarely do they help to decide what task to do next). In deciding what to do to data and how to do it, specifications received from users may prove insufficient, and certain system defaults may be inappropriate. Intelligent subsystems are needed to navigate such situations.

GIS users need help in whittling away options that aren't useful at given stages of their work, and could use advice on how to specify the options they decide to use. Attempts at providing such advice have been made using logic programming and expert system shells. Most applications of AI technology to GIS, however, attempt to automate data manipulations, such as deciding the size, format and placement of feature labels on maps (which may be the most popular testbed). This usually involves referring to a set of phenomena- and technique-based rules which are evaluated and weighed together to make tactical data processing decisions. Williams (1989) offers a good description how geographic (or any) expertise is cast into AI rules and tools:

> Intelligence can be achieved via two fundamental, but integrated sources. These sources are those of data relationships and structure, and techniques and procedures for manipulating and analyzing the data relationships. These sources can be considered as forming *expertise*. *Expertise* consists of knowledge about a particular domain (*real-world* geographic structures), understanding of the domain problems, and *skill* at solving some of these problems. Knowledge (in any speciality) is

usually of two sorts: *public* and *private. Public knowledge* includes the published definition, facts, and theories of which textbooks and references in the domain of study are typically composed. But *expertise* usually involves more than just this *public knowledge*. Human experts generally possess *private knowledge* that has not found its way into the published literature. This *private knowledge* consists largely of rules of thumb that have come to be called *heuristics. Heuristics* enable the human expert to make educated guesses when necessary, to recognize promising approaches to problems, and to check effectively with errorful or incomplete data. Elucidating and reproducing such *knowledge* is the central task of building *expert systems*. (Williams, 1989, p. 558)

There is little difference between providing expert advice to users and applying this knowledge to automated procedures. Deciding what-to-do-next and how-to-do-it can be informed by consulting expert systems; it is a matter of style whether users choose to adjudicate this information themselves or let software handle such decisions. Capacities to exercise these options should be built into toolkits provided to users. While providing such capabilities may increase the possibility of "use error" (Beard, 1989), it can also help to avoid them by advising users of the nature and consequences of their assumptions, choices and actions.

The native intelligence of a GIS is highly conditioned by what information it maintains to qualify data items. The more such metadata about objects, locations and attributes that a GIS maintains, the more confidently it can be used. But how many GIS's document the level of encoding (Boolean, nominal, ordinal, interval, ratio, etc.) used for a given variable? How many store, much less can interpret, units of measure (e.g., persons per square mile, hectares, pH, ppm, BTUs, furlongs per fortnight) for items they maintain? Information can be lost, misinterpreted or made spurious unless appropriate operations are applied to data items, which requires that variables be qualified by their levels and units of measure, properties rarely included in GIS data models and definitions. As a result, it may be difficult for GIS users to switch between English and Metric units in thematic maps or reports they generate, or be warned that attributes created by adding or subtracting ordinal values may be meaningless. Users may not be sure what the units of a variable are, whether a given datum represents a percent, a category or a scalar; analytic functions may not "know" that multiplying income per capita with population density yields units of income density. Ignoring data quality information generates wrong results and incurs missed opportunities. In most cases, the mechanisms for handling properly qualified data are rudimentary; perhaps the fact that it is inconvenient for procedural languages to include metadata in function arguments has begged the problem, leading us to believe that dealing with it requires artificial rather than innate intelligence.

What might help GIS software deal with data quality better than it now does? Procedures need some additional intelligence to enforce metadata-based constraints, and will need to reference a few new data structures in order to do this. Implementations should avoid both transfer of large data objects between storage units and storage of seldom-used fields. OOP architecture may help in this regard: attributes of data objects such as parcels or rivers can be implemented as instances of classes of data, each of which has specific levels and units of measure, spatio-temporal and accuracy measures, and rules and methods for their manipulation. Because the same attribute class may be shared by otherwise unrelated data objects, any OOP environment which handles metadata in terms of classes must support multiple inheritance (non-hierarchical object composition, as when describing the wetlands contained in a lot or the parcels occupying a swamp).

Summary and Conclusions

Geographic Information Systems have been a commercial commodity for nearly a decade. This technology has proven its value in land record systems, environmental impact analysis, facility management, land development, urban planning and other areas of application. Trained GIS users are in short supply, partly because so many of their skills are improvised and so much of the knowledge required to run GIS applications is undocumented. But as GIS databases grow larger and the features they encode become more complex, the pathways of spatial analysis tend to multiply and results become more equivocal. Unless we can bring more intelligence to bear, our GIS applications may not continue to yield as useful, interpretable and replicatable findings as we might wish.

To effectively apply geographic information systems, users need better tools for constructing, navigating and processing databases. Vendors cannot hope to provide users with "solutions" to very many geoprocessing problems in the form of full-featured and fully-documented applications. Rather, they should concentrate on adopting data models that better express spatial structure and process, software tools that extend both system functionality and ways to apply it, and advisory subsystems that can identify semantic subtleties when users query and analyze spatial-temporal data. GIS architecture will have to change at micro, meso and macro levels to enable these capabilities. New paradigms, unfamiliar data structures and relatively unproven techniques may have to be deployed. In the process, researchers, developers and users will have to alter their modes of operation. Geographic

Information Systems have already changed the ways in which we think about maps; the time has come to change how we think about, build and use this potent but imperfect technology.

References

Beard, Kate (1989). *Use error: The neglected error component.* **Proc. Autocarto 9**. Falls Church, VA: ACSM, pp. 808-817.

Berry, Brian (1964). *Approaches to regional analysis: a synthesis.* **Annals**, Assoc. of American Geographers, v. 54, pp 2-11.

Chrisman, Nicholas (1984). *The role of quality information in the long-term functioning of a geographic information system.* **Cartographica**, vol. 21, nos. 2&3, pp. 79-87.

Cooke, Donald (1989). *TIGER and the "Post-GIS" ERA.* **GIS World**, vol. 2 no. 4, ps. 40 *ff*.

Cooke, Donald and William Maxfield (1967). *The development of a geographic base file and its uses for mapping.* **Proc. URISA.**

Corbett, James (1977). *Topological principles in cartography.* **Proc. Autocarto II**. Falls Church, VA: ACSM, pp. 61-65.

Cox, Brad (1987). **Object Oriented Programming: An Evolutionary Approach.** Reading, MA: Addison-Wesley, 274 p.

Dutton, Geoffrey (1984). *Geodesic modelling of planetary relief.* **Cartographica**. vol. 21. nos. 2 & 3, pp. 188-207.

Dutton, Geoffrey (1989a). *Planetary modelling via hierarchical tessellation.* **Proc. Autocarto 9**. Falls Church, VA: ACSM, pp. 462-471.

Dutton, Geoffrey (1989b). *The Fallacy of Coordinates.* **Multiple Representations: Scientific report for the specialist meeting.** B. P. Buttenfield and J. S. DeLotto, eds. National Center for Geographic Information and Analysis, tech. paper 89-3, pp. 44-48.

Fekete, Gyorgy (1990). *Rendering and managing spherical data with Sphere Quadtrees.* **Proc. Visualization 90**. New York: ACM.

Goodchild, M. (1988). *Stepping over the line: Technical constraints and the new cartography.* **The American Cartographer**, vol. 15, no. 3, pp. 311-319.

Goodchild, Michael and Yang Shirin (1989). *A hierarchical data structure for global geographic information systems.* Santa Barbara, CA: National Center for Geographic Information and Analysis, tech. paper 89-5.

Kacmar, Charles (1989). *A process-oriented extensible hypertext architecture.* **ACM SIGCHI Bulletin**, vol. 21 no. 1, pp. 98-101.

Langran, Gail (1989). *Accessing spatiotemporal data in a temporal GIS.* **Proc. Autocarto 9**. Falls Church, VA: ACSM, pp. 191-198.

Peucker, Thomas and Nicholas Chrisman (1975). *Cartographic Data Structures.* **The American Cartographer**. vol. 1, pp. 55-69.

Peuquet, Donna (1988). *Issues involved in selecting appropriate data models for Global databases.* **Building Databases for Global Science**. H. Mounsey and R. Tomlinson (eds.). London: Taylor & Francis, pp. 187-260.

Sabella, P. and I. Carlbom, (1989). *An object-oriented approach to the solid modeling of empirical data.* **IEEE Computer Graphics & Applications**. vol.9, no. 5, pp. 24-35.

Tomlinson, Roger (1988). *The impact of the transition from analogue to digital cartographic representation.* **The American Cartographer**. vol. 15, no. 3, pp. 249-261.

Upson, Craig, Thomas Faulhaber, David Kamins, David Laidlaw, David Schlegel, Jeffery Vroom, Robert Gurwitz, Andries van Dam (1989). *The Application Visualization System: A computational environment for scientific visualization.* **IEEE Computer Graphics and Applications**. vol. 9, no. 4, pp. 30-42.

White, Denis and Jonathan Corson-Rikert (1987). *"WYSIWYG" map digitizing: Real time geometric and topological encoding.* **Proc. Autocarto 8**. Falls Church, VA: ACSM, pp. 739-743.

Williams, Robt. (1989). *Geographic Information: Aspects of phenomenology and cognition.* **Proc. Autocarto 9**. Falls Church, VA: ACSM, pp. 557-566.

CHARACTERIZATION OF FUNCTIONS REPRESENTING TOPOGRAPHIC SURFACES

Gert W. WOLF
Department of Geography, University of Klagenfurt
Universitaetsstrasse 65, A-9010 Klagenfurt, Austria
e-mail: WOLF@EDVZ.UNI-KLAGENFURT.ADA.AT

ABSTRACT

During the past years it has become apparent that a general framework of spatial data management resting on formal methods is indispensable. One aspect of such an analytic framework is the adequate characterization of functions so that they may be regarded as abstract models of real topographic surfaces. The importance of a precise mathematical description like this results from the fact that the theoretical requirements of differentiability and continuity of the derivatives, which are commonly employed in practical applications, do not suffice for functions to represent realizable topographic surfaces. The reason for this failure is that continuously differentiable mappings may still be endowed with some pecularities which are extremely unlikely to appear in reality and thus prevent the functions from being suitable models for the topography of a given area. It will be demonstrated that a great many of these pecularities are due to structural instability - a phenomenon which can easily be explained by the presence or absence of degenerate critical points and saddle connections. Since it can be proved that any function possessing degenerate critical points may be approximated accurately enough by another one without such points, mappings of the latter type (so-called Morse functions) which have, in addition, no saddle connections should describe topographic surfaces in an appropriate way. The results arrived at in this paper, however, are valid not only for functions defined on the plane but also for mappings defined on differentiable manifolds and thus help to diminish the deficiency of theoretical knowledge concerning curved surfaces as has been complained recently.

1.INTRODUCTION

As a consequence of the numerous applications of computers in cartography during the past years it has become apparent that a general framework of spatial data management and analysis is indispensable. This realization has given rise to an increasing number of publications concerning the formal foundations of numerous cartographic concepts. The different approaches covered a wide portion of the field of cartography ranging from the development of analytic tools for cartographic generalization (e.g. WOLF 1988a,b, 1989, WEIBEL 1989) to the design of databases for geographic information systems (e.g. PEUCKER 1973, PEUCKER/CHRISMAN 1975, PEUQUET 1983, BOUDRIAULT 1987, SALGÉ/SCLAFER 1989).

The subject of the present paper is the formal analysis of another point of interest in computer cartography, namely the adequate characterization of functions so that they may be regarded as abstract models of real topographic surfaces. The importance of a formal characterization like this is derived above all from the following four facts: First of all, theoretical results obtained for functions describing topography hold also for functions describing phenomena like population density, accessibility, pollution, temperature, precipitation etc.[1]; secondly, topographic surfaces represent the underlying continuous model of DTMs whereby DTM may stand as abbreviation for 'digital terrain model' or 'discrete terrain model' respectively; thirdly, a great many of the results derived for mappings from $R^2 \rightarrow R$ are also true for real-valued mappings defined on curved surfaces - so-called differentiable manifolds. As it has been pointed out just recently especially this point deserves our special attention '(since) geographical data (are) distributed over the curved surface of the earth, a fact which is often forgotton ...(However,) we have few methods for analyzing data on the sphere or spheroid, and know little about how to model processes on its curved surface ...' (GOODCHILD 1990, p.5f.). The final and perhaps the most important fact why topographic surfaces should be characterized in a formal way is that a formal characterization clearly reveals those concepts which are commonly used in practice but which are seldom or never explicitely stated.

2.TOPOGRAPHIC SURFACES

In almost any geographic or cartographic application functions $f(x,y)$ describing the topography of a given area and associating with each point (x,y) its respective altitude are presumed to be at least twice continuously differentiable. This concept, however, is just an ideal one since, for example, overhanging rocks imply that there is no definite correspondence between certain points and their altitudes or breaklines prevent $f(x,y)$ from being differentiable. In order to apply the powerful tool of calculus, nevertheless, the original concept has to be modified by assuming that the continuously differentiable functions are not the terrain itself but rather sufficiently close approximations of it[2] (cf. WOLF 1988a, 1990).

The question remaining, which seems to be deceptively simple in appearance but which, however, leads rather deeply into abstract mathematics is whether the theoretical requirements of differentiability and continuity of the derivatives suffice for functions to represent realizable topographic surfaces. As will be shown within the next chapters, this must not always be true because such mappings may be endowed with a number of pecularities like degenerate critical points or saddle connections which are extremely unlikely to appear in real-world applications and thus prevent the functions

[1] In order to achieve substantial results in non-topographic applications one will, however, have to ensure that data points are not too scarcely distributed.

[2] This supposition is also valid for mappings describing socio-economic, physical and other phenomena.

from being suitable models for the topography of a given area[3].

Since the detailed investigation of these pecularities requires several concepts from multidimensional calculus, it seems appropriate to repeat some basic definitions and theorems before continuing with the analysis of the addressed phenomena.

Definition 2.1 *A function (mapping) $f : R^n \rightarrow R$ is a rule associating with each $(x_1, x_2, \ldots, x_n) \in R^n$ a unique element $f(x_1, x_2, \ldots, x_n) \in R$.*

Though the previous definition has been given for the n-dimensional case we will restrict ourselves in most instances to two dimensions since this is the commonest case in practical applications. A further advantage of the restriction to functions $f(x, y)$ of only two variables is the fact that these mappings can be easily visualized, thus offering the chance to prefer a geometric approach rather than an abstract one. As a consequence, we will give - whenever possible - not only formal definitions but also geometric interpretations of the concepts being introduced. To start with, let us draw our attention to

Definition 2.2 *The partial derivative f_x of a function f(x,y) with respect to the variable x is the derivative of f with respect to x while keepig y constant. The partial derivative f_y of f with respect to y is defined in an analogous way. The partial derivatives evaluated at the particular point (x_0, y_0) are denoted by $f_x(x_0, y_0)$ and $f_y(x_0, y_0)$ respectively.*

Geometrically speaking, $f_x(x_0, y_0)$ specifies the tangens of the angle between the tangent to the intersecting curve $f(x, y_0)$ and the line $y = y_0$ parallel to the x-axis. To phrase it differently, $f_x(x_0, y_0)$ indicates the slope of the surface $f(x, y)$ at the point (x_0, y_0) in direction to the x-axis. It is hardly necessary to point out that $f_y(x_0, y_0)$ can be interpreted in a similar way.

Provided that $f(x, y)$ has partial derivatives at each point $(x, y) \in R^2$, then f_x and f_y are themselves functions of x and y which may also have partial derivatives. These second derivatives (derivatives of order two) are defined recursively by $(f_x)_x = f_{xx}$, $(f_x)_y = f_{xy}$, $(f_y)_x = f_{yx}$ and $(f_y)_y = f_{yy}$. For partial derivatives of order two the following theorem, which is important from a theoretical as well as from a practical point of view, holds[4].

Theorem 2.1 *If the partial derivatives f_{xy} and f_{yx} of a function f(x,y) are continuous in R^2 then $f_{xy} = f_{yx}$ in R^2.*

Partial derivatives of order higher than two are defined recursively in an analogous way. We will, however, desist from giving their exact definition since partial derivatives of first and second order are sufficient for the purpose of this paper. Instead we will turn our interest to another point which is of utmost importance for the following chapters and concerns the special arrangement of the partial derivatives of order two in form of a matrix, the so-called Hessian matrix.

[3] For the sake of simplicity we will illustrate these phenomena by examining mappings which are given explicitly and not in form of sparsely distributed data points in combination with an interpolation rule.

[4] For a proof cf. ENDL/LUH (1976, p.185f.).

Definition 2.3 *Let f(x,y) be a function whose partial derivatives* f_{xx}*,* f_{xy}*,* f_{yx} *and* f_{yy} *exist. The matrix* $Hf = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{pmatrix}$ *is termed the Hessian matrix of f.*
The Hessian matrix evaluated at a point (x_0, y_0) *is defined by* $\begin{pmatrix} f_{xx}(x_0,y_0) & f_{xy}(x_0,y_0) \\ f_{yx}(x_0,y_0) & f_{yy}(x_0,y_0) \end{pmatrix}$ *and denoted by* $Hf|_{(x_0,y_0)}$.
The determinant det(Hf) of the Hessian matrix Hf is called the Hessian determinant; when evaluated at the point (x_0, y_0) *it is denoted by* $det(Hf)|_{(x_0,y_0)}$.

With the aid of partial derivatives it is now possible to characterize those functions precisely which have been commonly employed for the approximation of topographic surfaces. These mappings are the so-called k-fold continuously differentiable functions whereby in almost any application a value of $k = 2$ has been chosen.

Definition 2.4 *A function f(x,y) is termed k-fold continuously differentiable, or of class* C^k*, if the partial derivatives up to order k exist and are continuous.*
A smooth function is a function of class C^∞*.*

3.NONDEGENERATE CRITICAL POINTS AND MORSE FUNCTIONS

Critical points[5] representing the peaks, pits and passes of surfaces play a major role not only in cartography but also in a great deal of other scientific applications where they represent either the extrema or the saddles of functions to be maximized or minimized. The importance of the critical points, which are also termed surface-specific points in computer cartography, for this field of research results from the fact that they contain significantly more information than any other point on the surface because they provide information about a specific location as well as about its surrounding (cf. PEUCKER 1973, PFALTZ 1976, PEUCKER/FOWLER/LITTLE/MARK 1978). As a consequence, their employment does not only ease the characterization and visual analysis of the topography of a given area but their application within digital terrain models also results in considerable savings in data capture and data management. Before stating two theorems which allow the classification of the critical points their formal description will be given.

Definition 3.1 *A point* (x_0, y_0) *is a (relative, local) maximum of f(x,y) if and only if* $f(x,y) < f(x_0,y_0)$ *for all* $(x,y) \in U_\epsilon(x_0,y_0)$.
A point (x_0, y_0) *is a (relative, local) minimum of f(x,y) if and only if* $f(x,y) > f(x_0,y_0)$ *for all* $(x,y) \in U_\epsilon(x_0,y_0)$.
A point (x_0, y_0) *is a saddle of f(x,y) if and only if f(x,y) has a local maximum along one line leading through* (x_0, y_0) *and a local minimum along another line leading through* (x_0, y_0).

[5] Unless stated otherwise critical points will be assumed to be nondegenerate.

According to the above definition saddle points are only those points with exactly two ridges (lines connecting passes with peaks) and exactly two courses (lines connecting passes with pits) emanating from them, thus excluding monkey saddles or the like. The following theorem[6] enables the computation as well as the classification of the critical points of a function $f(x, y)$ by applying the concepts of the partial derivatives and the Hessian determinant of $f(x, y)$.

Theorem 3.1 *(x_0, y_0) is a local maximum of a function f(x,y), which is twice continuously differentiable in R^2, if and only if $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$, $det(Hf)|_{(x_0,y_0)} > 0$ and $f_{xx}(x_0, y_0) < 0$ (or equivalently $f_{yy}(x_0, y_0) < 0$).*
(x_0, y_0) is a local minimum of a function f(x,y), which is twice continuously differentiable in R^2, if and only if $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$, $det(Hf)|_{(x_0,y_0)} > 0$ and $f_{xx}(x_0, y_0) > 0$ (or equivalently $f_{yy}(x_0, y_0) > 0$).
(x_0, y_0) is a saddle point of a function f(x,y), which is twice continuously differentiable in R^2, if and only if $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$ and $det(Hf)|_{(x_0,y_0)} < 0$.
(x_0, y_0) is a nondegenerate critical point of a function f(x,y), which is twice continuously differentiable in R^2, if and only if $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$ and $det(Hf)|_{(x_0,y_0)} \neq 0$.

An equivalent characterization of the critical points of a function $f(x, y)$ can be given by examining the eigenvalues of the corresponding Hessian matrix (cf. NACKMAN 1982, p.65 or NACKMAN 1984, p.444f.). The application of eigenvalues has moreover the advantage that they can also be used for the precise mathematical description of ridges, courses, flats, slopes as well as convex and concave hillsides of topographic surfaces (cf. LAFFEY/HARALICK/WATSON 1982, HARALICK/WATSON/LAFFEY 1983). We will, however, refrain from discussing all of these topographic phenomena since this would go far beyond the scope of the present paper.

Theorem 3.2 *Let f(x,y) be twice continuously differentiable in R^2 and $(x_0, y_0) \in R^2$. Further let $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$ and the determinant of the Hessian matrix Hf evaluated at (x_0, y_0) be unequal to zero. Then there is*
a (local) maximum at (x_0, y_0) if the number of negative eigenvalues of $Hf|_{(x_0,y_0)}$ is two,
a saddle at (x_0, y_0) if the number of negative eigenvalues of $Hf|_{(x_0,y_0)}$ is one and
a (local) minimum at (x_0, y_0) if the number of negative eigenvalues of $Hf|_{(x_0,y_0)}$ is zero.

The number of negative eigenvalues of $Hf|_{(x_0,y_0)}$ is also termed the index of (x_0, y_0); thus a maximum is a critical point of index two, a saddle is a critical point of index one, and a minimum is a critical point of index zero. The so-defined index of a critical point may be also interpreted as an 'index of instability (since) a ball displaced slightly from a relative minimum "will roll back" to that minimum. It is a point of stable equilibrium; ... A ball displaced from a saddle point may or may not return to that point of equilibrium, depending on the direction of displacement; while a ball displaced from a

[6] A proof can be found in any standard book on elementary calculus as e.g. in COURANT (1972, p.159f.).

relative maximum is completely unstable' (PFALTZ 1976, p.79, cf. also PFALTZ 1978, p.7f.)[7].

Another advantage of the employment of the eigenvalues of Hf is the fact that this concept may be transfered to n−dimensional differentiable manifolds which represent generalizations of the Euclidean n−space. Formally a manifold is characterized by

Definition 3.2 *An n-dimensional topological manifold is a separable*[8] *metric space in which each point has a neighbourhood homeomorphic*[9] *to R^n.*

An n−dimensional manifold thus represents nothing more than a topological space with the same local properties as the Euclidean n−space. MASSEY (1967, p.1) gives a vivid illustration of the two-dimensional case of this analogy when describing an intelligent bug crawling on a surface (two-dimensional manifold) and being unable to distinguish it from a plane (R^2) due to his limited range of visibility.

The previously defined manifolds, however, must be given some additional structure so that the concept of differentiability has meaning, thus yielding to differentiable manifolds. For the sake of simplicity and because only the concept itself is needed we will drop their formal definition[10] and imagine them as something looking like R^n but being smoothly curved. Examples of two-dimensional differentiable manifolds are the sphere or the torus whereas the cube, the cone or the cylinder are none. With differentiability being specified for mappings defined on manifolds[11] it is possible to investigate not only functions defined on the plane (R^2) but moreover mappings defined on surfaces (two-dimensional manifolds) as e.g. functions describing the distribution of precipitation over the globe because a lot of theoretical results for such mappings can be easily obtained due to the homeomorphic relationships between differentiable manifolds and Euclidean space[12]. Thus the concept of a differentiable manifold as it has been sketched above offers the chance to diminish the deficiency of theoretical knowledge concerning curved surfaces as it has been complained by GOODCHILD (1990, p.5f.) and to counteract his criticism.

Since practice has shown that degenerate critical points are extremely unlikely to occur in real-world applications, functions possessing exclusively

[7] An intuitive classification of the critical points according to their degree of (un)stability has been given by PEUCKER (1973, p.28f.) and WARNTZ/WATERS (1975, p.485f.).

[8] In a topological space, a set $A \subseteq B$ is dense in a set B if $\bar{A} = B$. A topological space C is termed separable if some countable set is dense in C.

[9] Two topological spaces A and B are called homeomorphic if there exists a bijective function $f : A \to B$ such that both f and f^{-1} are continuous.

[10] The precise mathematical characterization of a differentiable manifold can be found e.g. in GAULD (1982, p.54) or PALIS/de MELO (1982, p.4).

[11] For an adequate definition cf. GAULD (1982, p.60).

[12] For example, it is possible to characterize the critical points by their partial derivatives, to make a distinction between degenerate and nondegenerate ones as well as to classify the latter into maxima, saddles and minima according to the number of negative eigenvalues of the associated Hessian matrix.

nondegenerate critical points have been studied comprehensively by numerous authors[13].

Definition 3.3 *A smooth function is termed a Morse function if all of its critical points are nondegenerate.*

For Morse functions the folllowing four theorems, whose importance will become obvious in the next chapters where the concepts of structural stability and the problem of approximating functions possessing degenerate critical points by mappings without such points will be discussed, hold (cf. ARNOL'D 1972, p.83, PFALTZ 1976, p.83, GAULD 1982, p.118, PALIS/de MELO 1982, p.89, FOMENKO 1987, p.80f.):

Theorem 3.3 *Each Morse function on a compact manifold has only a finite number of critical points; in particular, all of them are distinct.*

Theorem 3.4 *The critical points of a Morse function are always isolated*[14].

Theorem 3.5 *The set of Morse functions is open and dense in the set of all k-fold differentiable functions defined on a manifold.*

Theorem 3.6 *Let f be a Morse function, which is defined on a simply-connected domain bounded by a closed contour line, then the number of minima of f minus the number of saddles of f plus the number of maxima of f equals two.*

The concept of Morse functions - though not explicitly mentioned - has been employed in almost every geographic application since they represent - with one restriction, which will be discussed in Chapter five - the prototype of mappings eligible to characterize topographic surfaces. One exception, however, constitutes the work of PFALTZ (1976, 1978) whose graph theoretic model for the characterization and generalization of topographic surfaces is based explicitly on attributes of Morse functions and thus represents the first attempt to describe those mappings formally which may be regarded as abstract models of the topography of a given area.

4.STRUCTURAL STABILITY

Modern philosophy of science requires that natural science accepts only those theories which can be verified at any time. As a consequence of this metatheoretical view the concept of repeatability saying that the same experiment must give the same result under the same conditions has become fundamental in modern sciences although, strictly speaking, the idea is just an ideal one. Ideal, because it is never possible to guarantee exactly the same conditions by abandoning all external factors even in the most carefully designed experiment. To an even greater extent one is confronted with the

[13] A great deal of the theoretical work is due to Morse (cf. MORSE/CAIRNS 1969).

[14] A critical point is called isolated if sufficiently close to it there exists no other critical point.

problem of repeatability in sciences like geography or cartography. For example, physical-geographic theories are based on measurements taken outside where side-effects are much less controllable than in the physicists' laboratories, or digital terrain models rest on digitized data which are affected by errors due to machine inaccuracy and/or human intervention.

Since the rigorous interpretation of repeatability would make any scientific work impossible the previously described idealized concept has been weakened by tolerating small changes in the conditions under which an experiment is carried out provided that these changes do not affect the result significantly. To phrase it differently, 'what we really expect is not that if we repeat the experiment under precisely the same conditions we will obtain precisely the same results, but rather that if we repeat the experiment under approximately the same conditions we will obtain approximately the same results. This property is known as structural stability ...' (SAUNDERS 1982, p.17). Mathematically, deviations from the ideal experiment which are caused by external factors are represented by perturbation functions and structural stability is the insensitiveness of the mapping or the familiy of mappings describing the experiment to these perturbation functions. The impact of this concept of structural stability for geography and cartography is that in these disciplines questions like the following ones have to be answered: Is a function describing a geographic phenomenon insensitive to small measurement errors and thus structurally stable? Is a family of functions describing a geographic phenomenon over time insensitive to temporal changes and thus structurally stable? Is a mapping representing the underlying continuous model of a digital terrain model insensitive to measurement errors and thus structurally stable?

When using the term 'structural stability', however, one has to distinguish between 'structural stability of a function' (cf. POSTON/STEWART 1978, p.63) and 'structural stability of a family of functions' (cf. POSTON/ STEWART 1978, p.92f., SAUNDERS 1982, p.17f.). In the above-mentioned geographic and cartographic applications of 'structural stability' the first interpretation of the term applies to the first and third examples while the second interpretation applies to the second example. Since in the present paper only the concept of a structurally stable function is of importance we will confine ourselves to this aspect of structural stability and proceed with an example in order to explain it[15].

Let us consider the functions $f_1(x) = x^2$ and $f_2(x) = x^2 + \varepsilon x$ with εx representing a perturbation function. For the derivatives of $f_1(x)$ and $f_2(x)$ holds:

$$\begin{aligned} f_1(x) &= x^2 & \qquad f_2(x) &= x^2 + \varepsilon x \\ f_1'(x) &= 2x & \qquad f_2'(x) &= 2x + \varepsilon \\ f_1''(x) &= 2 & \qquad f_2''(x) &= 2 \end{aligned}$$

[15] For the sake of simplicity we will restrict ourselves thereby to functions of a single variable.

In order to obtain the critical points of the two functions we set the first derivatives to zero, solve the resulting equations with respect to x and examine the second derivates which represent the one-dimensional analogue of the Hessian matrix.

$$\begin{aligned} 2x &= 0 \\ x &= 0 \\ f_1''(0) &= 2 > 0 \end{aligned} \qquad\qquad \begin{aligned} 2x + \varepsilon &= 0 \\ x &= -\frac{\varepsilon}{2} \\ f_2''(-\frac{\varepsilon}{2}) &= 2 > 0 \end{aligned}$$

The above calculations indicate that the perturbation function moves the minimum from $x = 0$ to $x = -\frac{\varepsilon}{2}$ in a way depending smoothly on ε (with ε being an arbitrary small number). The type of the critical point, however, as well as the structure of the graph of $f_1(x)$ in a surrounding of $x = 0$ are not affected by the perturbation (see also Fig. 4.1) and therefore the function is structurally stable at $x = 0$.

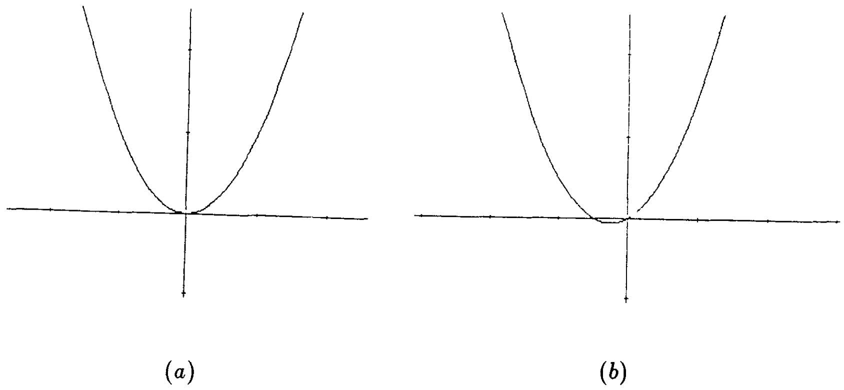


Fig. 4.1 Graphs of the functions (a) $f_1(x) = x^2$ and (b) $f_2(x) = x^2 + \varepsilon x$.

Next let us examine the two mappings $g_1(x) = x^3$ and $g_2(x) = x^3 + \varepsilon x$ with εx representing again a perturbation function. For the derivatives of $g_1(x)$ and $g_2(x)$ holds:

$$\begin{aligned} g_1(x) &= x^3 \\ g_1'(x) &= 3x^2 \\ g_1''(x) &= 6x \end{aligned} \qquad\qquad \begin{aligned} g_2(x) &= x^3 + \varepsilon x \\ g_2'(x) &= 3x^2 + \varepsilon \\ g_2''(x) &= 6x \end{aligned}$$

In order to determine the critical points of $g_1(x)$ and $g_2(x)$ we again set the first derivatives to zero, solve the resulting equations with respect to x

and inspect the second derivatives.

$$\begin{aligned} 3x^2 &= 0 \\ x &= 0 \\ g_1''(0) &= 0 \end{aligned} \qquad\qquad \begin{aligned} 3x^2 + \varepsilon &= 0 \\ x_{1,2} &= \pm\sqrt{-\frac{\varepsilon}{3}} \\ g_2''(\pm\sqrt{-\frac{\varepsilon}{3}}) &= \pm 6\sqrt{-\frac{\varepsilon}{3}} \end{aligned}$$

The previous calculations yield to the following interesting result: While the function $g_1(x)$ has a degenerate critical point at $x = 0$, the mapping $g_2(x)$ - which is obtained from $g_1(x)$ by adding the term εx - has no critical points for positive ε but two critical points, namely a local minimum at $x_1 = +\sqrt{-\frac{\varepsilon}{3}}$ and a local maximum at $x_2 = -\sqrt{-\frac{\varepsilon}{3}}$, for negative ε thus showing an irregular unstable behaviour. Illustrations of the function $g_2(x) = x^3 + \varepsilon x$ for different values of ε are depicted in Fig. 4.2.

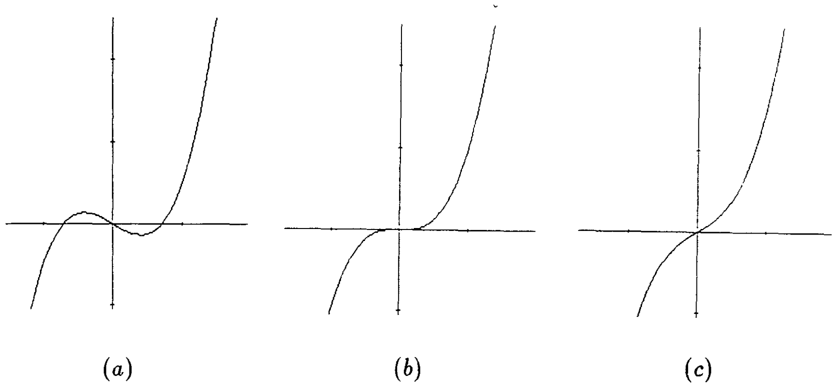


Fig. 4.2 Graphs of the function $g_2(x) = x^3 + \varepsilon x$ for (a) $\varepsilon < 0$, (b) $\varepsilon = 0$ and (c) $\varepsilon > 0$.

The different behaviour of the two functions $f_1(x)$ and $g_1(x)$ in a surrounding of $x = 0$ can be explained by the following theorem (cf. POSTON/STEWART 1978, p.63f.).

Theorem 4.1 *A critical point is structurally stable if and only if it is nondegenerate.*

In the above example $f_1(x)$ has a nondegenerate critical point at $x = 0$ while $g_1(x)$ has a degenerate one at this location. In the first case, as a consequence of the structural stability induced by the nondegenerate critical point the perturbation function does not change the type of the point but only moves its location. In the second case, however, the degeneracy of the critical point causes structural instability resulting in a change of the type of the critical point.

A direct consequence of the last theorem is the following one which shows once more the importance of Morse functions (cf. POSTON/STEWART 1978, p.70f.).

Theorem 4.2 *Morse functions are structurally stable.*

At the beginning of this chapter the importance of structural stability for scientific work in general has been indicated. At this point its importance for geography and cartography will be demonstrated in the light of some possible applications. The most essential one will certainly concern those fields in geographic and cartographic research where the systems approach is already well established as e.g. in ecology, climatology, demography etc. and functions or families of functions are used to describe ecosystems, weather, population dynamics etc. The major question to be answered in the above examples is whether a given system is stable or unstable over time and in the latter case if it will explode or collapse. A second field of applications comprises the analysis of functions describing cartographic and geographic phenomena like terrain, population density, accessibility, temperature and the like. The question to be answered in this context is which data points are best selected so that the functions obtained are structurally stable. It can be assumed, however, that those mappings that are derived from surface-specific points which are taken to be nondegenerate will produce results being superior to all others. Finally, a third point worth mentioning in this connection is the analysis of structural stability due to measurement errors caused by machine inaccuracy and/or human intervention - a problem which will have to be tackled in combination with the aid of statistics.

5.DEGENERATE CRITICAL POINTS AND SADDLE CONNECTIONS

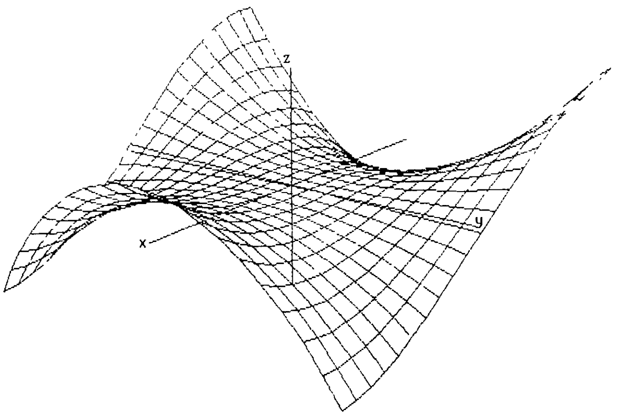
Degenerate critical points form - besides saddle connections - part of those phenomena which prevent continuously differentiable mappings from being suitable models for the topography of a given area. The reason is that degenerate critical points are - according to Theorem 4.1 - structurally unstable and thus unlikely to appear in real-world applications since any perturbation would immediately destroy them.

Formally, a degenerate critical point (x_0, y_0) is characterized by the fact, that the partial derivatives $f_x(x_0, y_0)$ and $f_y(x_0, y_0)$ as well as the Hessian determinant $det(Hf)|_{(x_0,y_0)}$ are zero. Some examples of functions possessing degenerate critical points are depicted in Fig. 5.1.

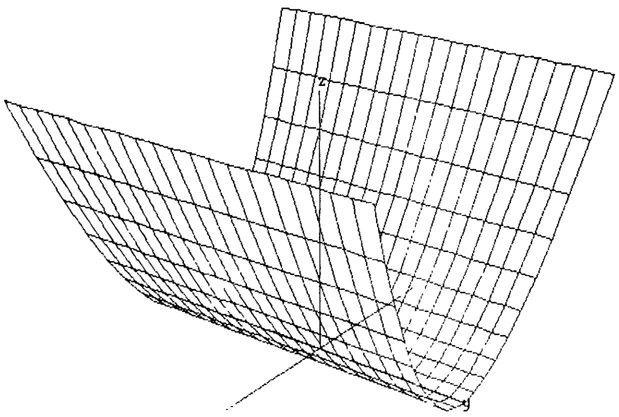
Though its definition sounds deceptively simple, degeneracy is a multifaceted phenomenon with different levels to be distinguished. A first subdivision can be made into isolated degenerate critical points and non-isolated ones with the latter being extremely uncommon[16] (cf. POSTON/STEWART

[16] For this reason they are excluded from further consideration.

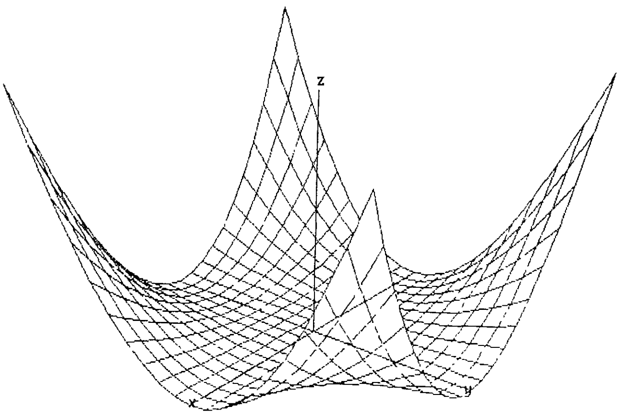
1978, p.53.). From the above sketched functions $f(x,y)$ has an isolated degenerate critical point at (0,0) while $g(x,y)$ has non-isolated ones along the x–axis and $h(x,y)$ along the x– and the y–axes. Besides this subdivision of the critical points another one can be made according to their degree of degeneracy (cf. FOMENKO 1987, p.80) which is explained next.



(a)



(b)



(c)

Fig. 5.1 Some functions having degenerate critical points: (a) $f(x,y) = x^3 - 3xy^2$ (monkey saddle at (0,0)); (b) $g(x,y) = x^2$ (pig-trough with degenerate critical points occuring along the x–axis); (c) $h(x,y) = x^2y^2$ (crossed pig-trough with degenerate critical points occuring along the x– and the y–axes).

Definition 5.1 *The degree of degeneracy of a critical point* (x_0, y_0) *is equivalent to the number of zero eigenvalues of* $Hf|_{(x_0,y_0)}$.

To illustrate this concept let us examine the two functions $f(x,y) = x^3 - 3xy^2$ and $g(x,y) = \frac{x^3}{3} - \frac{y^2}{2}$. By setting the first partial derivatives to zero and inspecting the second partial derivatives it can be shown that both $f(x,y)$ and $g(x,y)$ possess a degenerate critical point at (0,0).

$$\begin{aligned} f(x,y) &= x^3 - 3xy^2 \\ f_x(x,y) &= 3x^2 - 3y^2 \\ f_y(x,y) &= -6xy \\ f_{xx}(x,y) &= 6x \\ f_{xy}(x,y) = f_{yx}(x,y) &= -6y \\ f_{yy}(x,y) &= -6x \end{aligned} \qquad \begin{aligned} g(x,y) &= \frac{x^3}{3} - \frac{y^2}{2} \\ g_x(x,y) &= x^2 \\ g_y(x,y) &= -y \\ g_{xx}(x,y) &= 2x \\ g_{xy}(x,y) = g_{yx}(x,y) &= 0 \\ g_{yy}(x,y) &= -1 \end{aligned}$$

The Hessian matrices of the two mappings run therefore

$$Hf = \begin{pmatrix} 6x & -6y \\ -6y & -6x \end{pmatrix} \qquad Hg = \begin{pmatrix} 2x & 0 \\ 0 & -1 \end{pmatrix}$$

and when evaluated at the critical point (0,0)

$$Hf|_{(0,0)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad Hg|_{(0,0)} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}$$

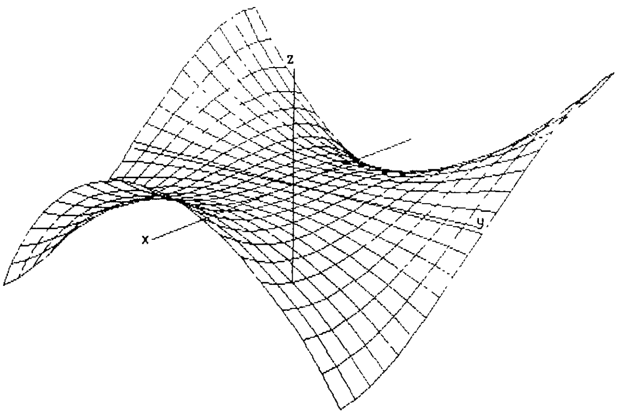
In order to obtain the eigenvalues of the two matrices we solve the corresponding characteristic polynoms

$$(0-\lambda)(0-\lambda) = 0 \qquad (0-\lambda)(-1-\lambda) = 0$$

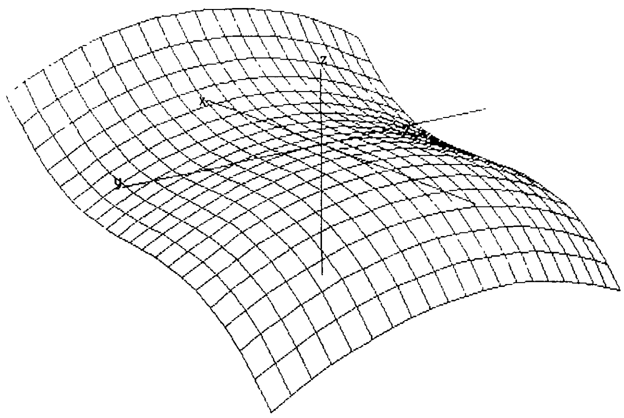
yielding

$$\lambda_{1,2} = 0 \qquad \begin{aligned} \lambda_1 &= 0 \\ \lambda_2 &= -1 \end{aligned}$$

Thus, in the first case the number of zero eigenvalues and therefore the degree of degeneracy of the critical point (0,0) is two, whereas in the second case the degree of degeneracy of (0,0) is one. Illustrations of the two functions in a surrounding of this location can be found in Fig. 5.2.



(a)



(b)

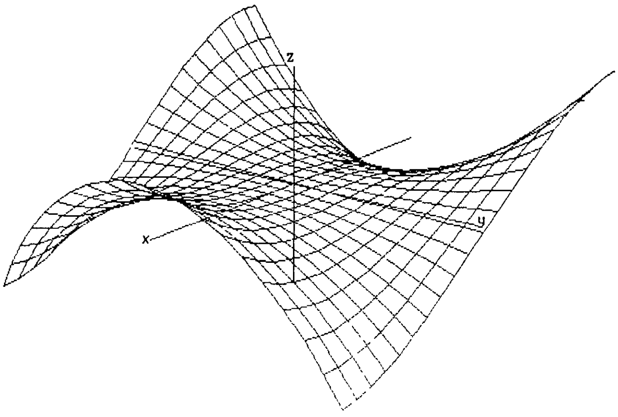
Fig. 5.2 Graphs of the functions (a) $f(x,y) = x^3 - 3xy^2$ and (b) $g(x,y) = \frac{x^3}{3} - \frac{y^2}{2}$.

Another point of interest concerning degeneracy is the question whether functions possessing degenerate critical points can be approximated accurately enough by mappings without such points, or in other words if it is possible to substitute degenerate critical points by nondegenerate ones. In order to answer this question let us recall that degenerate critical points are structurally unstable and that the set of Morse functions is open and dense in the set of all differentiable mappings defined on a manifold. It can be proved that due to these two properties the question asked earlier can be answered affirmatively since the following theorem[17] holds.

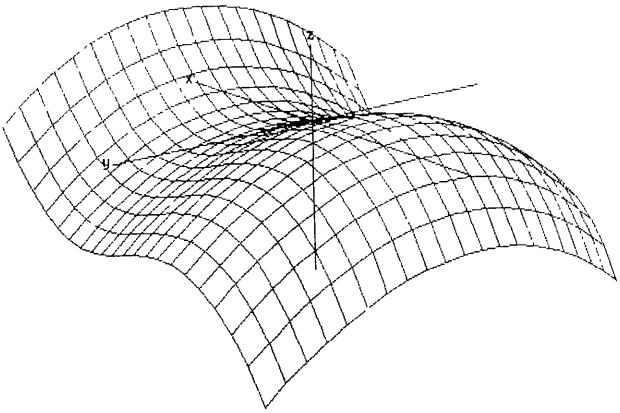
Theorem 5.1 *If a function has a degenerate critical point, then by an arbitrarily small shift of the function it can be ensured that the complicated singularity is dispersed into several nondegenerate ones.*

The above theorem, however, does not provide any information about the number nor about the types of the nondegenerate critical points one obtains when splitting a degenerate one. The following examples illustrate two possible cases that might occur when mappings are interfered by perturbation functions by means of the two mappings $f(x,y) = x^3 - 3xy^2$ and $g(x,y) = \frac{x^3}{3} - \frac{y^2}{2}$ both possessing a degenerate critical point at (0,0) (see Fig. 5.2). Deformations of $f(x,y)$ and $g(x,y)$ by the perturbation functions εy and εx respectively yield $\tilde{f}(x,y) = x^3 - 3xy^2 - \varepsilon y$ and $\tilde{g}(x,y) = \frac{x^3}{3} - \frac{y^2}{2} - \varepsilon x$. When

[17] An exact proof of this theorem which is rather complicated and requires several concepts like transversality, jet-spaces etc. from such branches of abstract mathematics as differential topology or catastrophy theory can be found in ARNOL'D (1972, p.65).



(*a*)



(*b*)

Fig. 5.3 Graphs of the functions (a) $\tilde{f}(x,y) = x^3 - 3xy^2 - \varepsilon y$ and (b) $g(x,y) = \frac{x^3}{3} - \frac{y^2}{2} - \varepsilon x$.

setting the first partial derivatives of $\tilde{f}$ and $\tilde{g}$ to zero, solving the resulting systems of equations with respect to x and y, and inspecting the second partial derivatives it becomes apparent that the degenerate critical point (0,0) of $f(x,y)$ is substituted by two nondegenerate saddles with locations at $(\sqrt{\frac{\varepsilon}{6}}, -\sqrt{\frac{\varepsilon}{6}})$ and $(-\sqrt{\frac{\varepsilon}{6}}, \sqrt{\frac{\varepsilon}{6}})$ respectively, whereas the degenerate critical point (0,0) of $g(x,y)$ is substituted by a nondegenerate saddle at $(\sqrt{\varepsilon}, 0)$ and a local maximum at $(-\sqrt{\varepsilon}, 0)$. The visualization of the effect of approximating a function having a degenerate critical point by a mapping without such points can be achieved by comparing Fig. 5.2 and Fig. 5.3 with the latter depicting the graphs of $\tilde{f}(x,y)$ and $\tilde{g}(x,y)$.

It can easily be demonstrated that degenerate critical points are not the only phenomena inducing structural instability but saddle connections will cause it, too. However, it has been proven that saddle connections may always be broken up by perturbation functions which have to be chosen in a convenient way (cf. GUCKENHEIMER/HOLMES 1983, p.60ff.). As a consequence of this result and Theorem 5.1 it can be concluded that Morse functions without saddle connections are the most suitable mappings to describe topographic surfaces because, on the one hand, they possess only structural stable elements like nondegenerate critical points but are, on the other hand, also eligible to approximate accurately enough structural unstable elements like degenerate critcal points and saddle connections.

6.CONCLUSION

In the present paper the characterization of those mappings which may be regarded as abstract models of topographic surfaces has been attempted. The importance of a characterization like this is derived from the fact that differentiability and continuity of the derivatives do not suffice for functions to represent realizable topographic surfaces because continuously differentiable mappings may nevertheless be endowed with pecularities which are unlikely to appear in reality. An analysis of these pecularities, however, reveals that they are primarily due to structural instability of the respective functions - a phenomenon induced by degenerate critical points or saddle connections. Therefore it has been concluded that mappings describing the topography of a given area should be Morse functions without saddle connections. It should be emphasized, however, that the results obtained in this article represent only the first step in the formal characterization of the topography of a given area because a great deal of important phenomena like junctions of channels and ridges have not been considered. The analysis of these phenomena and its incorporation into a general framework of spatial data management will have to be the subject of future research.

7.REFERENCES

ARNOL'D, V. I., 1972. Lectures on Bifurcations in Versal Families. In: Russian Mathematical Surveys, p.54 - 123.

BOUDRIAULT, G., 1987. Topology of the TIGER File. In: Proceedings of AUTOCARTO 8, p.258 - 263.

COURANT, R., 1972. Vorlesungen ueber Differential- und Integralrechnung 2. Berlin, Heidelberg, New York.

ENDL, K./LUH, W., 1976. Analysis II. Wiesbaden.

FOMENKO, A. T., 1987. Differential Geometry and Topology. New York, London.

GAULD, D. B., 1982. Differential Topology. An Introduction. New York, Basel.

GOODCHILD, M. F., 1990. Spatial Information Science. In: Proceedings of the Fourth International Symposium on Spatial Data Handling, p.3 - 12.

GUCKENHEIMER, J./HOLMES, P., 1983. Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. New York, Berlin, Heidelberg, Tokyo.

HARALICK, R. M./WATSON, L. T./LAFFEY, T. J., 1983. The Topographic Primal Sketch. In: The International Journal of Robotics Research, p.50 - 72.

LAFFEY, T. J./HARALICK, R. M./WATSON, L. T., 1982. Topographic Classification of Digital Image Intensity Surfaces. In: Proceedings of the IEEE Workshop on Computer Vision, Theory and Control, p.171 - 177.

MASSEY, W. S., 1967. Algebraic Topology: An Introduction. New York, Chicago, San Francisco, Atlanta.

MORSE, M./CAIRNS, S. S., 1969. Critical Point Theory in Global Analysis and Differential Topology. New York, London.

NACKMAN, L. R., 1982. Three-dimensional Shape Description Using the Symmetric Axis Transform. Ph.D. dissertation. Department of Computer Science, University of North Carolina, Chapel Hill.

NACKMAN, L. R., 1984. Two-Dimensional Critical Point Configuration Graphs. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, p.442 - 450.

PALIS, Jr. J./de MELO W., 1982. Geometric Theory of Dynamical Systems. New York, Heidelberg, Berlin.

PEUCKER, T. K., 1973. Geographic Data Structures. Progress Report After Year One. Technical Report #1, 'Geographic Data Structures' Project, ONR Contract N00014-73-C-0109.

PEUCKER, T. K./CHRISMAN, N., 1975. Cartographic Data Structures. In: The American Cartographer, p.55 - 69.

PEUCKER, T. K./FOWLER, R. J./LITTLE, J. J./MARK, D. M., 1978. The Triangulated Irregular Network. In: Proceedings of the American Society of Photogrammetry. Symposium on Digital Terrain Models (DTM), St. Louis, p.516 - 540.

PEUQUET, D., 1983. A Hybrid Structure for the Storage and Manipulation of Very Large Spatial Data Sets. In: Computer Vision, Graphics, and Image Processing, p.14 - 27.

PFALTZ, J. L., 1976. Surface Networks. In: Geographical Analysis, p.77 - 93.

PFALTZ, J. L., 1978. Surface Networks, an Analytic Tool for the Study of Functional Surfaces. Final report on NSF Grant OCR-74-13353.

POSTON, T./STEWART, I., 1978. Catastrophy Theory and Its Applications. London, San Francisco, Melbourne.

SALGÉ, F./SCLAFER, M. N., 1989. A Geographic Data Model Based on HBDS Concepts: The IGN Cartographic Data Base Model. In: Proceedings of AUTOCARTO 9, p.110 - 117.

SAUNDERS, P. T., 1982. An Introduction to Catastrophe Theory. Cambridge.

WARNTZ, W./WATERS, N., 1975. Network Representations of Critical Elements of Pressure Surfaces. In: Geographical Review, p.476 - 492.

WEIBEL, R., 1989. Konzepte und Experimente zur Automatisierung der Reliefgestaltung. Dissertation. Philosophische Fakultaet, Universitaet Zuerich.

WOLF, G. W., 1988a. Generalisierung topographischer Karten mittels Oberflaechengraphen. Dissertation. Institut fuer Geographie, Universitaet Klagenfurt.

WOLF, G. W., 1988b. Weighted Surface Networks and their Application to Cartographic Generalization. In: BARTH, W. (ed.), 1988. Visualisierungstechniken und Algorithmen. Berlin, Heidelberg, p.199 - 212.

WOLF, G. W., 1989. A Practical Example of Cartographic Generalization Using Weighted Surface Networks. In: DOLLINGER, F./STROBL, J. (eds.), 1989. Angewandte Geographische Informationstechnologie (= Salzburger Geographische Materialien, Heft 13), p.125 - 143.

WOLF, G. W., 1990. Metric Surface Networks. In: Proceedings of the Fourth International Symposium on Spatial Data Handling, p.844 - 856.

SIMULATION OF THE UNCERTAINTY OF A VIEWSHED

Peter F. Fisher,

Department of Geography,
Kent State University,
Kent, Ohio 44242-0001, U.S.A.
PFISHER1@KENTVM.BITNET

SUMMARY

One of the most widely available procedures packaged with GIS for the analysis of a Digital Elevation Model (DEM) is the identification of the viewable area, or the viewshed. The elevations recorded in the DEM do, however, contain error, and the USGS, for example, publishes a Root Mean Squared Error (RMSE) for each DEM. Research reported here assesses the uncertainty of locations being within a viewshed, given the published error for the DEM. In this research, repeated error fields are simulated with variable spatial autocorrelation, and added to the original DEM. The viewshed is then determined in the resulting noisy DEM. Results show that using the basic assumption of spatial independence in the error which is implicit in the RMSE remarkably few points are reliably within the viewshed. With spatially autocorrelated noise, the reliability is higher, but still should be cause for concern to many using viewshed procedures.

INTRODUCTION

Research on the propagation of error within GIS operations has focused upon the polygon overlay operation (MacDougall, 1975; Newcomer and Szajgin, 1984; Chrisman, 1989; Maffini et al., 1989; Veregin, 1989), at the expense of other GIS data types and functions. The experiments reported here examine one aspect of the propagation of error from a Digital Elevation Model (DEM) into the derivative product showing visible locations, sometimes known as a viewshed (see also Felleman and Griffin, 1990; Fisher, 1990).

This paper starts by briefly discussing the viewshed operation, and the nature of error in DEM data. The general methodology of simulating error is then discussed, followed by its application to a real location.

VIEWSHEDS AND DEMS

The basic algorithm in establishing the viewshed examines the line-of-sight between two points (the viewpoint and a target), and assesses whether any land or object rises above that line-of-sight. If it does then the target is not within the viewshed of the viewing location, but if no land rises above the elevation, then the target is within the viewshed. In establishing the viewshed either all possible targets in the area of a database (Clarke, 1990, 227-228), or only those within some constrained portion of the area (Aronoff, 1989, 234), may be considered. Several studies have explored differences in viewshed algorithms (Anderson, 1982; DeFloriani et al., 1986; Sutherland et al., 1974), and Felleman and Griffin (1990) have compared the output of four different GIS-based implementations of the viewshed operation. They show the viewsheds delimited to be very different. This difference is not particularly surprising given the multiple decisions to be made in designing the implementation of the viewshed operation. For example, decisions have to be made as to whether the viewpoint in a gridded DEM is anywhere within the viewpoint gridcell, or is just the mid-point; similarly, should the surface be treated as the stepped phenomena it is coded as, or an interpolated surface? The outcome of such algorithm-design decisions may produce dramatically different viewshed results in some DEMs.

The viewshed is invariably reported as a binary product, a target location is either within or without the viewshed of the viewpoint. No shades of uncertainty are admitted; neither the likelihood nor the probability of a point being within, or of being without the viewshed is reported. In the light of the considerable interest in database accuracy this seems remarkable, especially when each DEM is required to be accompanied by an error report (USGS, 1987).

The USGS has adopted the Root Mean Squared Error (RMSE) for reporting accuracy in their DEM products (USGS, 1987). The RMSE for any one DEM is based on the comparison between the elevations of at least twenty locations on the map, and their elevations recorded in the database. It should be noted that most USGS source maps are stated to conform to the National Map Accuracy Standards, which themselves state that "at no more than 10 percent of the elevations tested will contours be in error by more than one half the contour interval", as established by comparison with survey data (Thompson, 1988, p 104). In generating a DEM from a map, therefore, at least two stages are present when error may be introduced: map compilation and DEM generation from the map. The error reported for the DEM only refers to the second of these, and it is only that error that is examined here. Some DEMs are generated directly from aerial photographs by the Gestalt Photo Mapper II, and, in this case, the error may only be introduced in one stage.

Error in DEMs is then widely acknowledged, and has been the subject of some study. That study has, however, concentrated on the nature and description of the error, not its propagation into any derivative products. The only work known to the current author which provides any evaluation of error propagation is by Felleman and Griffin (1990). They have compared implementations of the viewshed operation, and simulated error in the DEM before calculating the viewshed, as is reported here. They examined 3 viewpoints in 2 test areas for each of which 10 error simulations were run. Results are, however, only reported for one test location.

METHOD

SIMULATING ERROR

A Monte Carlo simulation and testing approach is taken to studying the propagation of DEM error here. In this approach, randomizing models of how error occurs are established, and then coded as computer procedures. The resulting computer program may be used to generate multiple realizations of the random process. Many workers have used original data in combination with realizations of the defined random process to establish the statistical significance of the original data with respect to the random process (Besag and Diggle 1977). Thus Openshaw et al. (1987) executed 499 realizations of the random process to locate two significant clusters of incidents of childhood leukemia in northern England.

How the error is distributed across the area of any one DEM is currently unknown, and factors that may effect the distribution of error is largely unresearched. The inference of the error reporting used by the USGS is that the error at any point occurs independently of that at any other point (i.e. the error is not spatially autocorrelated). Therefore, the following algorithm may be implemented (Fisher, in press):

```
1.   Define a standard deviation of a normal
     distribution (S = RMSE);

2.   Read Original_Value for the current cell:
     2.a  Using the Box-Muller (or some other)
          algorithm generate a random number drawn
          from a normal distribution with mean = 0
          and standard deviation = S;
     2.b  Add the random number to the
          Original_Value for the current cell, to
          give the New_Value;

3.   Repeat 2 for all cells in the Map_File.
```

This assumes that the standard deviation of a normal distribution is equivalent to the RMSE. In the absence of any other information on error structure, this may not be unreasonable. Such independent error is, however, very likely to contribute only a small portion of the overall error. High spatial autocorrelation is probably present, and banding can often be seen in the DEM data. To accommodate the occurrence of spatial autocorrelation, a version of the algorithm given by Goodchild (1980) was implemented, using Moran's I to measure the autocorrelation (Goodchild 1986; Griffith 1987). It works thus:

```
1.  Define a target autocorrelation ($I_t$), and a
    standard deviation of a normal distribution ($S = RMSE$);

2.  For each cell in the DEM generate a random
    value, with a normal distribution with mean = 0
    and standard deviation = S (see first
    algorithm);

3.  Calculate the Spatial Autocorrelation of the
    field ($I_1$);

4.  Randomly identify two cells in the DEM:
    4.a  Swap the values in the two cells;
    4.b  Calculate the new spatial autocorrelation
         ($I_2$);
    4.c  IF  $I_t > I_1$ AND $I_2 > I_1$ THEN retain the
             swap, and $I_1 = I_2$
         OR
         IF  $I_t < I_1$ AND $I_2 < I_1$ THEN retain the
             swap, and $I_1 = I_2$
         ELSE swap the two cells back to their
             original values;

5.  Repeat 4 until ($I_t - I_1$) is within some
    threshold.

6.  For each cell in the original DEM, add the
    value in the corresponding autocorrelated
    field.
```

This algorithm is simple and can be made computationally efficient, and it will be noted is an extension of the first algorithm listed.

The random number generator used in programming the algorithms was also tested, since like all such implementations it is truly a pseudo-random number generator (Ripley, 1986). The generator included with Turbo Pascal 5.5 was used here. The runs test was used to check for serial autocorrelation, the chi-squared test was used to check for a uniform distribution, and serial autocorrelation was tested for all lags to check for cycling in the generator. The generator performed satisfactorally for all cases, when number sequences up to 10,000 long were tested (corresponding to the 100 by 100 array used in the generation of autocorrelation).

MEASURING UNCERTAINTY

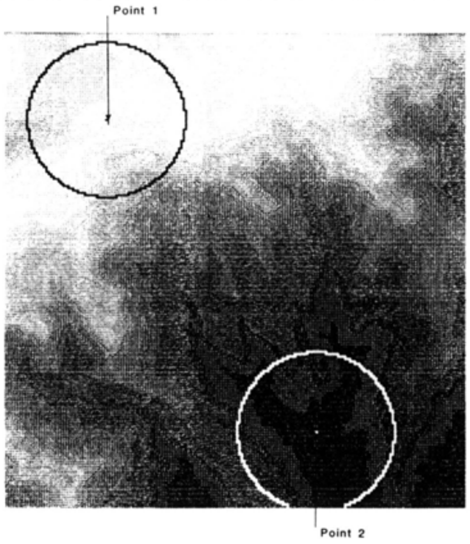
Each realization of the random process then resulted in a new DEM. The viewshed was calculated for each, and for any one view point, all the viewsheds in which the noise term had the same RMSE and spatial autocorrelation were summed to yield a summation image which describes the uncertainty of the viewshed, or the fuzzy viewshed. Since the viewshed is reported as a raster image coded as 0 or 1, the maximum value in the fuzzy image is 20, the number of realizations. It is possible to define the viewshed with a particular likelihood (probability). Thus, with 20 realizations, cells with value 19 in the resulting image, have probability, $p = 19/20 = 0.95$ of being within the viewshed, and, similarly, those with value 10 have probability, $p = 10/20 = 0.5$.

THE STUDY AREA

A 200 x 200 cell subset of the USGS Prentiss, NC, 7.5 minute DEM was acquired covering the Coweeta Experimental Watershed (Fig. 1). This DEM has been the subject of considerable research on DEM products (Band, 1986; Lammers and Band, 1990). Within the area of the DEM two test viewing locations (viewpoints) were arbitrarily

Figure 1

The Digital elevation model of the Coweeta Experimental Watershed, N.C. The two test locations are shown, and the 1 km zone around each indicated.



identified, one near an interfluve (Point 1), and one in a valley bottom (Point 2). All viewsheds calculated in the research reported here were only to within 1km of the viewpoint, and from an elevation of 2m above the viewpoint (corresponding to approximately the near-view, and the eye level of an individual, respectively).

The DEM was read into a format compatible with Idrisi (Eastman, 1989), a PC based package for Geographic analysis, and all further processing was done with either Idrisi modules, or implementations of the above algorithms written by the author in Turbo Pascal version 5.5. The VIEWSHED module of the Idrisi package is crucial to the research reported here, and so some simple test situations were established to examine the veracity of the viewable area calculated by that module. In every test, the module performed satisfactorally. The module operates on a DEM of any size, by using random access files, but at great expense in processing time. Only examining locations within 1 km of the viewpoint also made the processing time required for the research realistic.

RESULTS

Tables 1 and 2 report the frequencies of occurrence of values in the fuzzy viewsheds derived from the noisy DEMs, and those fuzzy viewsheds are shown in Figures 2 and 3. For each set of viewsheds with a specific spatial autocorrelation in the noise, and for a particular viewpoint, the tables record in the first column the frequencies of cells which are outside the viewshed in the original DEM, but inside those in simulated elevation models, the second column records those that are in both viewsheds, and the last column records the sum of the first two. The results all refer to applications of noise with variable spatial autocorrelation and with RMSE = 7, the value specified for this DEM. Table 1 and Figure 2 show results for Point 1, while Table 2 and Figure 3 show results for Point 2.

DISCUSSION

It is apparent in both Tables 1 and 2 that when there is no spatial autocorrelation in the noise, there are very low frequencies of cells with high cell counts in the viewsheds of either test viewpoint. 8 and 9 cells occur within all 20 of the viewsheds of the two points (i.e. the nearest neighboring cells plus 1 in 1 case), and in the case of Point 2 only 16 cells have cell counts of 18 or greater. The viewshed of the higher, ridge-top location (Point 1) appears to be more stable, however, with higher frequencies of cells with count greater than 10 (giving $p > 0.5$ of being within the viewshed), 706 as opposed to 443 for Point 2.

TABLE 1

Frequencies of occurrence of values between 1 and 20 in the image resulting from summing all noisy viewsheds, for Point 1 where the autocorrelation in the noise varies from 0 to 0.9. All points within 1 km of the viewpoint are included.

	I=0			I=0.7			I=0.9		
Cell Count	Out View	In View	Sum	Out View	In View	Sum	Out View	In View	Sum
0	1456	1	1457	1425	1	1426	1574	3	1577
1	192		192	226	3	229	204	2	206
2	142	9	151	123	1	124	100	3	103
3	122	18	140	117	3	120	68	6	74
4	92	31	123	81	9	90	72	16	88
5	76	45	121	90	14	104	47	15	62
6	60	60	120	63	30	93	35	29	64
7	37	78	115	47	49	96	35	32	67
8	21	97	118	23	66	89	34	35	69
9	15	123	138	21	87	108	31	49	80
10	3	120	123	4	73	77	14	48	62
11	6	94	100	6	73	79	5	57	62
12	4	74	78	3	78	81	7	61	68
13	3	93	96		104	104	3	63	66
14		80	80		84	84		73	73
15		75	75		85	85		90	90
16		90	90		93	93		82	82
17		81	81		86	86		110	110
18		73	73		123	123		92	92
19		24	24		142	142		101	101
20		9	9		71	71		308	308

The distribution of cell count frequencies becomes progressively less skewed towards the low frequencies as the autocorrelation in the noise increases. Indeed, in the case of Point 1, the distribution becomes strongly bimodal when I = 0.9. When the noise perturbing the DEM has high autocorrelation, the frequency of cells with high counts increases, so that as the value of I for the noise increases the number of cells with count 20 increases dramatically for both Point 1 (9, 71, and 308 for I = 0, 0.7 and 0.9), and Point 2 (8, 38, and 112). There are, however, only slight, but probably useful, rearrangements of frequencies in many of the other cell counts, and an increase in the number of cells with only a count of 1 can be noted in the case of Point 1. At Point 2, the number of cells with count 1 is reduced by nearly a third, but the number of cells with count 20 does not increase by nearly as much as in the results for Point 1. There is, however, an evening of frequencies corresponding to counts from 5 to 20, which is not observed in the results for Point 1.

As the spatial autocorrelation increases the number of cells that are identified as not even possibly being within the viewshed, but within the search distance of the viewpoint, does increase with autocorrelation, but the change is not continuous in both cases. From Point 1, the counts are 1456, 1425 and 1574 when I = 0, 0.7, and 0.9 respectively, and at Point 2, the values are 1109, 918, and 897. Furthermore, the number of cells that may be within the viewshed (>0 in the fuzzy viewshed) but were not in the viewshed in the original DEM, increases with autocorrelation at Point 2, (822, 1013, and 1034 for I = 0, 0.7, and 0.9 respectively), but at Point 1 the reverse is true (733, 804, and 655 respectively). The upper frequencies of cells outside the original viewshed changes very little either between or within viewpoints (frequencies of 12 to 15 can be noted).

TABLE 2

Frequencies of occurrence of values between 1 and 20 in the image resulting from summing all noisy viewsheds, for Point 2 where the autocorrelation in the noise varies from 0 to 0.9. All points within 1 km of the viewpoint are included.

	I=0			I=0.7			I=0.9		
Cell Count	Out	In View	Sum	Out	In View	Sum	Out	In View	Sum
0	1109	16	1125	918	3	921	897	1	898
1	319	33	352	265	12	278	255	2	257
2	179	89	268	205	21	226	177	6	183
3	122	90	212	143	33	176	137	13	150
4	73	119	192	100	46	146	124	19	143
5	46	122	168	83	63	146	88	29	117
6	47	132	179	64	81	145	60	39	99
7	14	152	166	42	76	118	64	59	123
8	8	137	145	45	85	130	45	66	111
9	5	135	140	27	97	124	39	89	128
10	1	85	86	25	101	126	25	113	138
11	5	99	104	10	109	119	9	88	97
12	1	82	83	4	116	120	4	100	104
13	1	78	79		107	107	3	115	118
14		64	64		106	106	4	113	117
15	1	42	42		120	120		119	119
16		36	36		97	97		111	111
17		18	18		93	93		109	109
18		7	7		91	91		125	125
19		1	1		49	49		117	117
20		8	8		38	38		112	112

SPATIAL ARRANGEMENT OF UNCERTAINTY

The spatial distribution of these fuzzy values are shown in Figures 2 and 3, together with the viewshed in the original DEM, the elevation map of the immediate area, and a viewshed image derived from the fuzzy image where I

Figure 2

DEM and Viewsheds of Point 1: A) the elevations in the near-view; B) the viewshed image from the original DEM; fuzzy viewsheds where C) I=0, D) I=0.7, and E) I=0.9; and F) an image showing the viewshed where p>=0.5 from E.

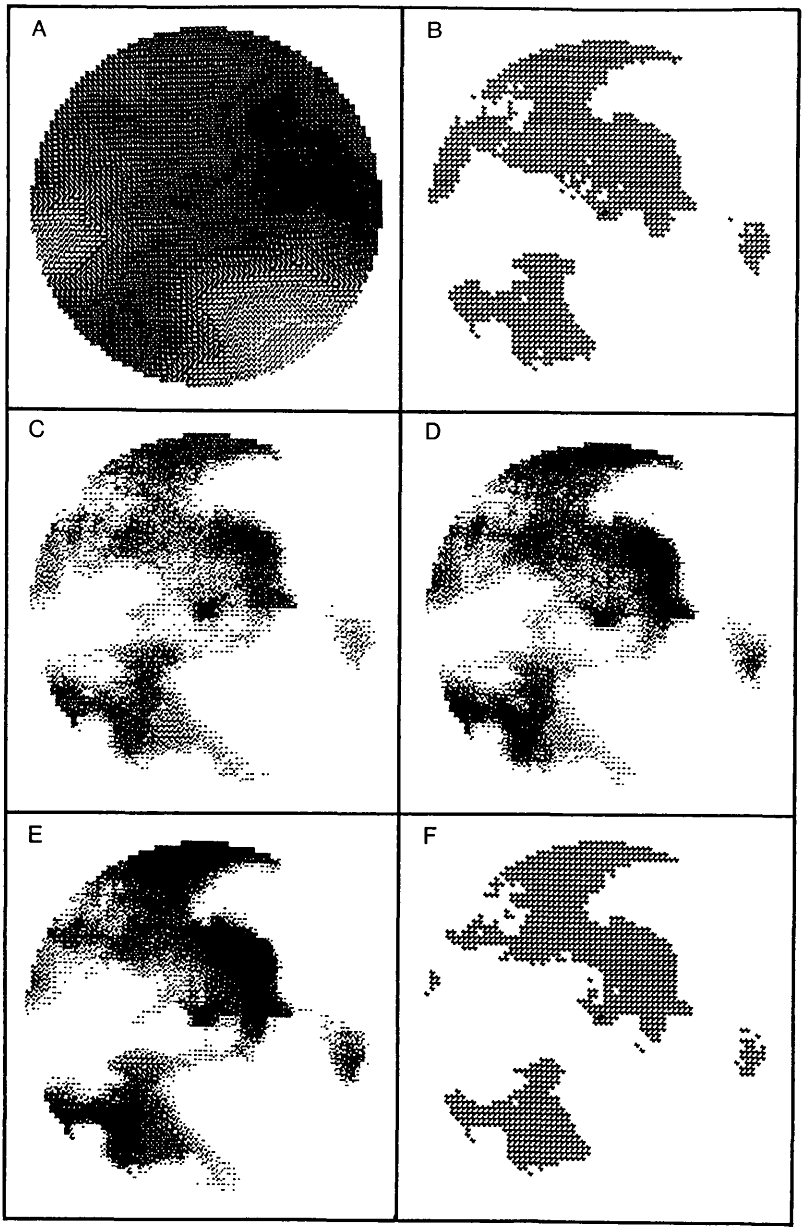
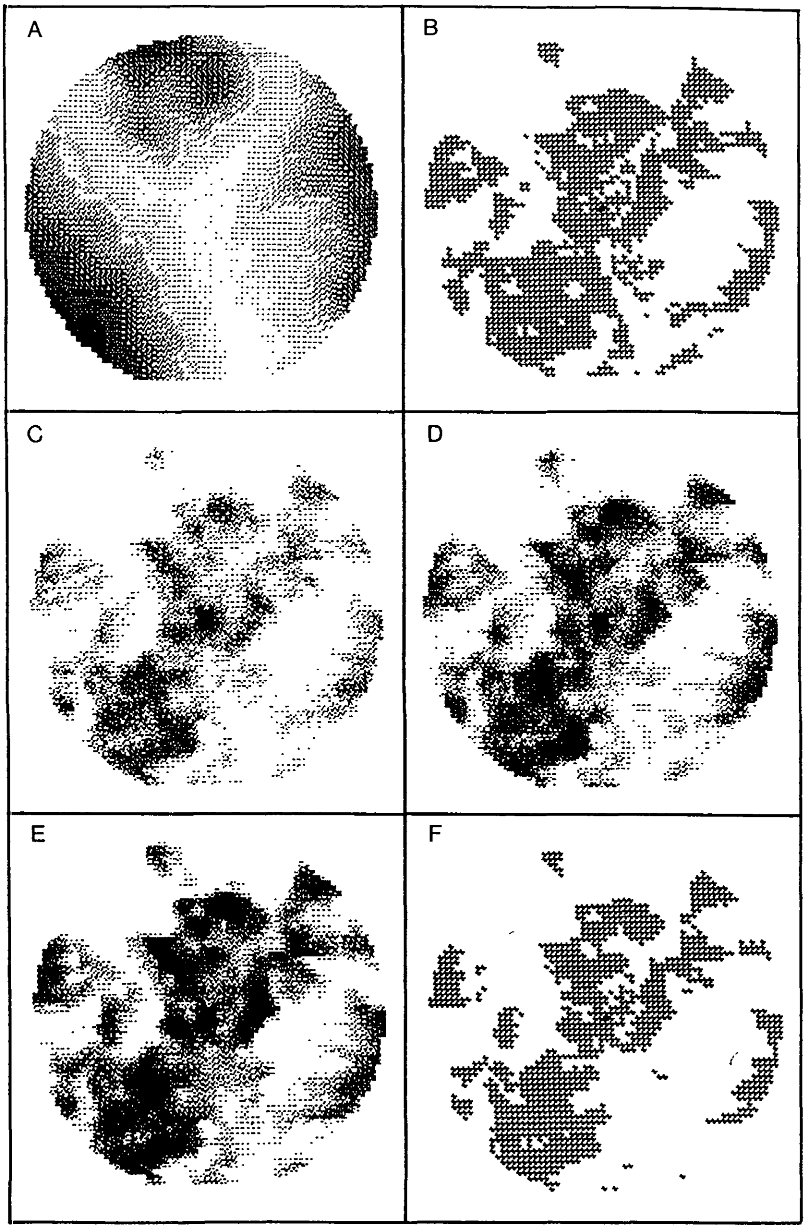


Figure 3

DEM and Viewsheds of Point 2: A) the elevations in the near-view; B) the viewshed image from the original DEM; fuzzy viewsheds where C) I=0, D) I=0.7, and E) I=0.9; and F) an image showing the viewshed where p>=0.5 from E.



in the error = 0.9, and value in the fuzzy viewshed >= 10 (probability of being within the viewshed >= 0.5). In those figures therefore, the spatial arrangements of the tabulations presented and discussed above can be seen.

In both areas, the application of increasing autocorrelation to the noise progressively increases the certainty of similar areas being within the viewshed. Thus the nucleus of the zones with high probability identifiable in those viewsheds where the noise term had I = 0.9 are identifiable in images where the noise had I = 0.

The ridge-top location, Point 1 (Fig. 2), has a higher frequency of high counts in the image. The areas of high likelihood of belonging to the viewshed are more contiguous for this location than for Point 2 (see Fig. 3); most of the high likelihood values are in three blocks of land, one immediately to the northeast of the viewpoint, one to the north, and the other to the southwest. From Point 2 (Figure 3), the areas of greater certainty are by contrast highly disjoint, although one large area does exist to the southwest.

Particularly, it should be noted that in neither test location is it possible to identify those areas that are of high likelihood in the fuzzy images from properties of the viewshed as calculated in the original DEM (Fig 2b, and 3b). For example, elevations both above and below the viewpoint may contain both high and low certainty.

CONCLUSION AND CONTINUING WORK

Firstly, it is possible to observe that no absolute certainty can be placed on the viewshed. Depending on the spatial autocorrelation that is applied to the noise term, it is apparent that the likelihood of cells being in the viewshed, can be very low. Indeed, with the assumption of spatial independence (where I = 0, the only assumption that is acceptable given the method of calculation and publication of the USGS error statement) very little if any certainty can be placed upon the standard viewshed calculated. Fortunately, perhaps, the viewshed from an elevated location seems to be more reliable than one in a depression, but work presented here is only exploratory.

Although the method used here is too computationally intensive for widespread implementation, it does yield alternative fuzzy viewsheds from a particular viewpoint. In this paper alternative fuzzy viewsheds derived from simulated DEMs with variable spatial autocorrelation are discussed. The algorithms can already accommodate variable RMSE, and can be recoded to accommodate variability in other parameters.

This paper points to further work in three main areas, the effect of the error, the source of it, and terrain control on the stability of the viewshed. In the first of these, it is necessary to develop a method to predict the fuzzy viewshed, which derives results similar to those generated by simulation, but which is more computationally efficient, and so possible to use in regular GIS operations. To achieve this it is necessary to explore further, probably by simulation, the relationships between error structures in DEMs, and fuzziness in viewsheds. The effects in the middle and far view should also be explored. In the area of error sources, considerable need exists for more information on the structure of the error in the DEM, and the relationship between error derived from digitization (the only error studied here), and that derived from original map compilation. Finally, aspects of relative elevation, other relief properties, and further aspects of terrain on patterns of fuzziness and the nature of error need to be explored.

ACKNOWLEDGEMENT

I wish to thank Larry Band, University of Toronto, who supplied the digital elevation data when it was needed, and Ron Eastman, Clarke University, for allowing me to examine the code of the Idrisi Viewshed module. John Felleman, Mike Goodchild, Dave Mark, and Dan Griffith all commented on aspects of the research, and Audrey Clarke assisted and advised on figure preparation. The work was conducted on a 386 PC, and I also wish to thank all those students who also wished to use it for their patience.

REFERENCES

Anderson, D.P., 1982, Hidden line elimination in projected grid surfaces: ACM Transactions on Graphics, Vol.1, pp.274-291.

Aronoff, S., 1989, Geographic Information Systems: A Management Perspective, WDL Publications, Ottawa.

Band, L.E., 1986, Topographic partition of watersheds with digital elevation models: Water Resources Research, Vol.22, pp.15-24.

Besag, J. and P.J.Diggle, 1977, Simple Monte Carlo tests for spatial patterns: Applied Statistics, Vol.26, pp.327.

Chrisman, N.R., 1989, Modeling error in overlaid categorical maps, in M.F.Goodchild and S.Gopal (Eds.), The Accuracy of Spatial Databases, Taylor and Francis, London, pp.21-34.

Clarke, K.C., 1990, Analytical and Computer Cartography, Prentice Hall, Englewood Cliffs, NJ.

DeFloriani, L., Falcidieno, B., and Pienovi, C., 1986, A visibility-based model for terrain features, in Proceedings of the Second International Symposium on Spatial Data Handling, pp. 235-250.

Eastman, R., 1989, IDRISI: A Grid-based Geographic Analysis System, Version 3.1, Graduate School of Geography, Clark University, Worcester, Mass.

Felleman, J., and Griffin, C., 1990, The role of error in GIS-based viewshed determination - A problem analysis, US Forest Service, North Central Experiment Station, Agreement No. 23-88-27.

Fisher, P.F., 1990, Simulation of Error in Digital Elevation Models, Papers and Proceedings of the 13th Applied Geography Conference.

Goodchild, M.F., 1980, Algorithm 9: Simulation of autocorrelation for aggregate data, Environment and Planning A, Vol.12, pp.1073-1081.

Goodchild, M.F., 1986, Spatial Autocorrelation, CATMOG 47, Geo Books, Norwich, UK.

Griffith, D.A., 1987, Spatial Autocorrelation: A Primer, Association of American Geographers, Washington, DC.

Lammers, R.B. and L.E.Band, 1990, Automating object representation of drainage basins, Computers and Geosciences, Vol.16, pp.787-810.

MacDougall, E.B., 1975, The accuracy of map overlays: Landscape Planning, Vol. 2, pp.23-30.

Maffini, G., Arno, M., and Bitterlich, W., 1989, Observations and comments on the generation and treatment of error in digital GIS data, in M.F.Goodchild and S.Gopal (Eds.), The Accuracy of Spatial Databases, Taylor and Francis, London, pp.55-67.

Newcomer,, J.A., and Szajgin,J., 1984, Accumulation of thematic map errors in digital overlay analysis: The American Cartographer, Vol. 11, pp.58-62.

Openshaw, S., M.Charlton, C.Wymer and A.Craft, 1987, A Mark 1 Geographical Analysis Machine for the automated analysis of point data sets: International Journal of Geographical Information Systems, Vol. 4, pp.335-358.

Ripley, B.D., 1987, Stochastic Simulation, John Wiley, New York.

Sutherland, I.E., Sproull, R.F., and Scumaker, R.A., 1974, A characterization of ten hidden-surface algorithms, Computing Surveys, Vol. 6, pp.1-55.

Thompson, M.M., 1988, Maps for America, Third Edition, USGS, Reston, Va.

USGS, 1987, Digital elevation models, Data Users Guide 5, USGS, Reston Va.

Veregin, H., 1989, Error modeling for the map overlay operation,iIn M.F.Goodchild and S.Gopal (Eds.), The Accuracy of Spatial Databases, Taylor and Francis, London, pp.3-18.

Dynamic Maintenance of Delaunay Triangulations

Thomas Kao*
David M. Mount†
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland

Alan Saalfeld
Bureau of the Census

Abstract

We describe and analyze the complexity of a procedure for computing and updating a Delaunay triangulation of a set of points in the plane subject to incremental insertions and deletions. Our method is based on a recent algorithm of Guibas, Knuth, and Sharir for constructing Delaunay triangulations by incremental point insertion only. Our implementation features several methods that are not usually present in standard GIS algorithms. Our algorithm involves:

Incremental update: During point insertion or deletion only the portion of the triangulation affected by the insertion or deletion is modified.

Randomized methods: For triangulation building or updates involving large collections of point, randomized techniques are employed to improve the expected performance of the algorithm, irrespective of the distribution of points.

Persistence: Earlier versions of the triangulation can be recovered efficiently.

1 Introduction

The Voronoi diagram and its dual, the Delaunay triangulation, are among the most useful structures that can be derived from a finite set of n points in the plane. These structures have long been recognized as being very useful in automated cartographic applications [6, 8]. Although it is known that these structures can be computed in worst case $O(n \log n)$ time [2, 5], it is widely felt that the implementation of these algorithms involves a significant amount of programming effort. As a consequence many implementors have settled for a simple incremental algorithm, which builds the

*The work of this author was partially supported by the Bureau of the Census under grant JSA 89-32.

†The work of this author was partially supported by the Bureau of the Census under grant JSA 89-32 and by the NSF under grant CCR-8908901.

diagram site by site [3, 5]. Although there are instances in which this algorithm runs in $O(n^2)$ worst case time, it is often observed that the performance of the incremental algorithm is rarely as bad as this quadratic bound suggests.

Recently Guibas, Knuth, and Sharir have given a theoretical explanation of this phenomenon [4]. They analyzed the complexity of the simple incremental algorithm for Delaunay triangulations combined with an novel technique for locating the triangle of the triangulation which contains a given point. They showed that, irrespective of the distribution of points, this algorithm operates in $O(n \log n)$ expected time provided that the points are inserted in random order. (Here the expectation is over the possible insertion orders.) We extend their result by giving an algorithm which can incrementally maintain a Delaunay triangulation through a sequence of insertions as well as deletions.

An incremental algorithm is said to have an *amortized time complexity* of $f(n)$ if the total cost of any sequence of N operations divided by N is $O(f(n))$, even though a single operation may have cost much greater than $f(n)$. We show that, given a base set of points, any sequence of insertions and deletions to the Delaunay triangulation can be performed on-line in expected amortized time $O(\log n)$ per insertion or deletion under the assumptions that (1) for insertion, each of the base points not present in triangulation is equally likely to be inserted, and (2) for deletion, each of the points present in the triangulation is equally likely to be deleted. Here n reflects the number of points present in the triangulation at the time of the update. No assumptions are made about the distribution of the base points.

Our algorithm has an interesting type of *persistence* property. In particular, we are able to reconstruct any earlier version of the triangulation more efficiently than the naive method of simply reversing the recent history of insertions and deletions.

We have implemented our algorithms in order to establish the actual efficiency, which was established theoretically by Guibas, Knuth and Sharir. We present a number of observations on the algorithm and its practical performance, and in particular we consider how the algorithm performs when the assumption of random insertion and deletion is violated.

The remainder of the paper is organized as follows. In Section 2 we describe the incremental insertion of Guibas, Knuth, and Sharir (for the sake of completeness). In Section 3 we describe the deletion algorithm and analyze its expected case complexity and in Section 4 we consider the complexity of sequences of insertions and deletions and how to keep the search structure balanced through such a sequence. In Section 5 we discuss our implementation of the algorithm and provide a number of graphs displaying the essential elements of the algorithm which determine its complexity.

2 Incremental Insertion

In this section we review the basic incremental algorithm as presented by Guibas, Knuth, and Sharir [4]. The algorithm is quite simple. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points and let $D(P)$ denote the Delaunay triangulation of this point set. For points $a, b, c \in P$, let $\triangle abc$ denote the triangle (not necessarily in the triangulation) determined by these points. We consistently label the vertices of triangles in counterclockwise order. For simplicity, we make the usual general position assumptions that no three points are colinear and that no four points are cocircular. These assumptions are handled in our implementation, but we omit discussion of them here since they clutter the presentation with undue detail.

We assume that the point coordinates have been normalized so that they lie within the interior of the unit square. It is assumed that the four corners of the unit square are always elements of P, and these four points are not eligible for deletion. When the algorithm is initiated, the Delaunay triangulation consists of two triangles formed by adding a diagonal through the unit square. (These four points actually violate our general position assumption, implying that either of the two diagonals could be used.)

The insertion procedure operates as follows. Suppose that p is a new point to be added to the triangulation. By a point location method (to be described later) determine the triangle $\triangle abc$ of $D(P)$ which contains this point. Replace the triangle $\triangle abc$ with three triangles $\triangle pab$, $\triangle pbc$, and $\triangle pca$ (see Fig. 1(a)). This operation is called *augmentation.*

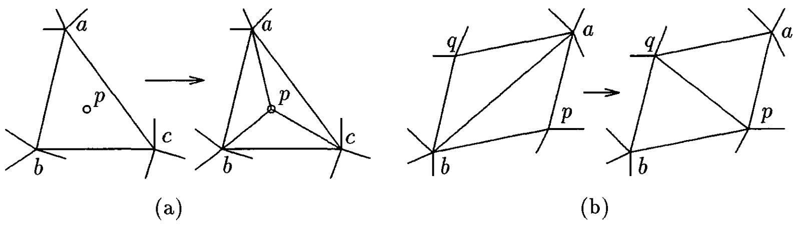


Figure 1: Incremental point insertion.

To determine whether each of these three new triangles, say $\triangle pab$, is a Delaunay triangle we perform the following Delaunay test. Let $\triangle qba$ be the triangle on the "other side" of the edge $\overline{ab}$. If either (1) no such triangle exists (because edge $\overline{ab}$ is an edge of the unit square) or (2) if the triangle does exist but p does not lie within the circumcircle of $\triangle qba$, then $\triangle pab$ is Delaunay and no further updating is needed. Otherwise replace the two triangles $\triangle pab$ and $\triangle qba$ with the two triangles $\triangle paq$ and $\triangle pqb$ (see Fig. 1(b)). This is equivalent to swapping the edges $\overline{ab}$ and $\overline{pq}$, and hence is called an *edge swap.* Continue the test, this time with the triangles $\triangle paq$ and $\triangle pqb$ until all triangles pass the Delaunay test. The Delaunay test is then performed for the for the other two triangles $\triangle pbc$ and $\triangle pca$. The correctness of this algorithm is well known (see, e.g. [5]).

Pseudocode for this algorithm is given below. Guibas, Knuth, and Sharir actually describe an elegant nonrecursive implementation of this algorithm [4]. We have presented the algorithm in this recursive form to emphasize its symmetry with the incremental deletion algorithm, which we present in the next section. The procedure invokes the primitive in(u, v, w, p) which determines whether the point p lies within the circumcircle of the triangle $\triangle uvw$ (where u, v and w are given in counterclockwise order). The argument p is the point to be inserted, and D is the existing triangulation.

```
**procedure** Insert($p, D$);
**begin**
    Find the triangle $\triangle abc$ of $D$ containing $p$;
    Replace $\triangle abc$ by the three triangles $\triangle pab$, $\triangle pbc$, and $\triangle pca$ in $D$;
    SwapTest($\overline{ab}$, D);
    SwapTest($\overline{bc}$, D);
```

```
        SwapTest($\overline{ca}$, D);
    end;

    procedure SwapTest($\overline{xz}$, D);
    begin
        if $\overline{xz}$ is an edge of the unit square then return;
        Let $y$ be the third vertex of the triangle to the right of $\overline{xz}$ in $D$;
        if in($x, y, z, p$) then begin
            Replace triangles $\triangle xyz$, $\triangle pxz$ with $\triangle pyz$, $\triangle pxy$ in $D$;
            SwapTest($\overline{xy}$, D);
            SwapTest($\overline{yz}$, D);
        end;
    end;
```

Letting (u_x, u_y) denote the coordinate of the point u, the primitive $\text{in}(u, v, w, p)$ is implemented by evaluating the following determinant.

$$\text{in}(u,v,w,p) \quad \equiv \quad \det \begin{pmatrix} u_x & u_y & u_x^2 + u_y^2 & 1 \\ v_x & v_y & v_x^2 + v_y^2 & 1 \\ w_x & w_y & w_x^2 + w_y^2 & 1 \\ p_x & p_y & p_x^2 + p_y^2 & 1 \end{pmatrix} > 0.$$

The data structure used for storing the triangulation can be chosen from any number of standard structures for storing subdivisions of the plane, such as the quad-edge data structure [5] or the winged-edge data structure [7]. These data structures are both edge-based in the sense that the primitive objects of the data structure are the edges. In our implementation a triangle-based data structure was employed. This is particularly convenient for the triangle-based point location techniques which discussed below. The fundamental property required of any data structure for this problem is that it be able to move from one triangle of the triangulation to each of its three neighboring triangles in constant time.

One important aspect of this algorithm is the particular order in which the triangles are deleted from the triangulation. Consider the set of triangles of the original triangulation which were replaced during insertion and let $R(p)$ denote the dual graph of this set of replaced triangles (where each vertex of this graph corresponds to a deleted triangle, and two vertices are adjacent if and only if these triangles share a common edge)

LEMMA 2.1 *The dual graph $R(p)$ is a tree. Further, if we take the root to be the triangle of the original triangulation which contains p, $\triangle abc$, then the sequence of deleted triangles forms a counterclockwise preorder traversal of this tree.*

PROOF: The dual graph is a tree because the union of the set of new triangles (those having p as a vertex) defines a simple polygon. (In fact this polygon is star-shaped with respect to p.) This polygon contains no other points of the point set in its interior. Thus the set of deleted triangles forms a triangulation of this polygon. It is well known that the triangulation of a simple polygon is a tree.

The fact that the deleted triangles define to a counterclockwise preorder traversal of this tree is an immediate consequence of the facts that (1) the edge swap is performed before either recursive call is made, and (2) the two recursive calls made in the algorithm are made in counterclockwise order relative to p. □

To analyze the complexity of the algorithm, Guibas, Knuth, and Sharir proved the following result [4]. Observe all of the triangles introduced by the insertion algorithm are adjacent to the new point p. Thus a triangle never "reappears" once it has been replaced (assuming insertions only).

THEOREM 2.1 (Guibas, Knuth, Sharir) *Let P be a set of n points in the plane, which are inserted in random order into a Delaunay triangulation using the above procedure. The expected number of triangles that appear at any time the construction is $O(n)$.*

Because the algorithm performs only a constant amount of work with each newly created triangle, it would follow that the expected running time of the algorithm is $O(n)$. However, the important missing element is the time required to determine which triangle the newly added point p lies in. It will be this point location problem which drives the total expected running time up to $O(n \log n)$. We describe two ways in which this point location can be performed.

The first method involves simple *bucketing.* Let us assume that the set of points P is known in advance. When the algorithm is initiated, the triangulation consists of a decomposition of the unit square into two triangles. We partition the initial point set into two groups, or *buckets*, depending on which triangle they lie in. As the triangulation is updated, we iteratively redistribute the points into finer and finer partitions, so that each triangle of the triangulation is associated with the set of points which lie within this triangle. (Our general position assumptions allow us to ignore the case in which a point lies on the edge of a triangle. In general this is handled by devising a rule which consistently forces all such points into one of the adjacent triangles. See also [5].) When a triangle is replaced by augmentation, only the points contained within this triangle need be rebucketed into one of three new triangles. When two triangles are replaced by two others through an edge swap, only the points in the original two triangles need be rebucketed into one of the two new triangles. (See Fig. 2(a).)

The second method was introduced by Guibas, Knuth, and Sharir. The *history* of the triangulation updates is stored. In particular, whenever a triangle $\triangle abc$ is replaced by two or more new triangles, $\triangle abc$ remains as part of the structure and marked as "old", and pointers are added from $\triangle abc$ to each of the newly generated triangles. The newly added triangles are called the *children* of the old triangles, and the old triangles are the *parents* of the new triangles. The number of children is either three (which occurs when an augmentation is performed) or two (which occurs when an edge swap is performed). Thus each node has a constant number of children.

Initially the data structure consists of a single node which implicitly represents the unit square (the only node which does not correspond to a triangle), and the insertion of the initial diagonal produces two triangular children. This process defines a rooted directed acyclic graph, which we call the *history graph.* The history graph is not a tree, because a given node may have as many as two parents in this structure (and the deletion algorithm of the next section may produce three parents).

In order to locate the triangle containing a newly added point, we start from root node representing the unit square, and trace through the chronological chain of "old" triangles containing this point until arriving at the triangle of the current triangulation which contains the point. At each "old" triangle there are at most three triangles at the next level which could contain the point, thus constant time suffices to determine the next triangle of the chain in which the point lies. (See Fig. 2(b).)

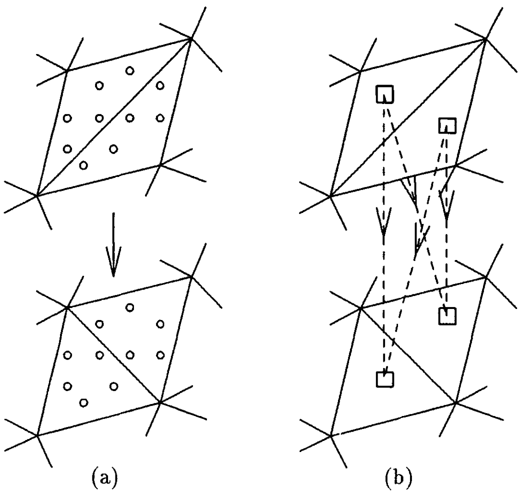


Figure 2: Point location.

Under the assumption that all points of the base set are inserted into the triangulation, the total time required by the bucketing and the history methods are identical, since the same discriminating tests are made for each point, and each point moves through the same sequence of triangles in each method. This history approach can be viewed as a sort of *lazy evaluation* of the bucketing scheme since it is only applied to the points which are indeed added to the triangulation. Thus if not all of the points are added to the final triangulation, the history method has an advantage over bucketing with respect to execution time. In addition this method need not know all the points in advance. The number of times a point is moved from one triangle to another can be as large as $O(n)$ per insertion. However, Guibas, Knuth, Sharir show that the number of triangles through which a point moves, when averaged over all the points and all insertions, is only $O(\log n)$ in the expected case. From this it follows that the incremental algorithm runs in $O(n \log n)$ time in the expected case, irrespective of whether the bucketing or history method is used.

One disadvantage of the history method is that its space usage is dependent on the number of edge swaps performed by the algorithm. Although this number is $O(n)$ in the expected case, it could be as large as $O(n^2)$ (although the probability of this occurring for large n is extremely small under the assumption of random insertion.) The bucketing method has the advantage that it never requires more than $O(n)$ space in the worst case even if the point insertion violates the randomness assumption. This is true because only the current triangulation is stored.

One big advantage of the history method is a type of *persistence.* Persistence refers to the ability of a data structure to maintain its history. In this case, by storing history of the data structure it is an easy matter to restore a recent version of the data structure. This is done quite simply by reversing the sequence of edge swaps by walking backwards through the history graph. Since the number of edge swaps per insertion is expected to be a constant (and we will see that the same holds true for deletion), the time needed to restore an earlier version of the triangulation is proportional to the number updates performed between the earlier version and the present one. This is a $\log n$ factor savings in running time over the naive method of reversing the string of recent operations. This same persistence will apply for deletions also as we shall see in the next section.

3 Incremental Deletion

In this section we introduce a simple incremental algorithm for deleting a point from a Delaunay triangulation. It seems inherently harder to implement a purely incremental deletion algorithm in the spirit of the insertion algorithm given in the previous section. Our deletion algorithm applies the insertion algorithm of the previous section in an off-line mode to compute an intermediate Delaunay triangulation, which it then uses to guide an incremental sequence of edge swaps to perform the actual deletion. Our algorithm has the interesting property that, with careful implementation (and assuming that points are in general position), it swaps edges in essentially the reverse order from the insertion algorithm. Thus, by calling the deletion algorithm on the points in the reverse order of insertion, the algorithm will incrementally disassemble the triangulation in exactly the reverse order of its assembly.

As before, let $P = \{p_1, p_2, \ldots, p_n\}$ denote a set of points in the plane (including the vertices of the unit square) and let $D(P)$ denote the Delaunay triangulation of this point set. Let $p \in P$ be the point to be deleted. We assume that p is not one of the vertices of the unit square. We make the same general position assumptions of the previous section that no three points are colinear and no four points are cocircular.

Let T denote the set of triangles incident to p in the Delaunay triangulation. Because p is not a vertex of the unit square, p does not lie on the convex hull of P, and hence the union of the triangles of T is a star-shaped polygon containing the point p in its interior (and in fact within its kernel). Let Γ denote this polygon. Observe that any triangle in T cannot be part of the triangulation after the deletion of p, and that any triangle in $D(P) - T$ (i.e. any triangle which is not incident to p) is still empty after the deletion of p. Thus, only the region of the plane covered by the polygon Γ need be retriangulated.

We begin by outlining a nonincremental algorithm, which we will shortly modify to give an incremental algorithm. By a cyclic enumeration of the triangles of T, determine the boundary vertices of the star-shaped polygon Γ. Compute the Delaunay triangulation D_Γ of the polygon Γ by any algorithm (see the remark below). Replace the triangles of T by the triangles of D_Γ giving the new Delaunay triangulation $D(P - \{p\})$.

Unfortunately, this algorithm is not incremental, and it is unclear how to modify the point location algorithms to deal with the sudden replacement of potentially $O(n)$ triangles by $O(n)$ new triangles. However, imagine that p were the last point of the triangulation to be inserted, prior to this deletion. The insertion of p would induce a particular sequence of edge swaps mapping $D(P - \{p\})$ to $D(P)$. Since we know both triangulations, it is a relatively simple matter to perform the edge swaps in reverse order to transform $D(P)$ incrementally to $D(P - \{p\})$.

We solve this problem by a simple leaf pruning method. Recalling the discussion preceding Lemma 2.1, a triangle of D_Γ is a *leaf* of the dual graph of D_Γ if and only if at least two of its sides lie on the boundary of Γ. From this lemma we know that by the preordering of replaced triangles, the edge swaps are performed in such an order that the leaves of Γ are the *last* triangles to be replaced in the triangulation. Thus,

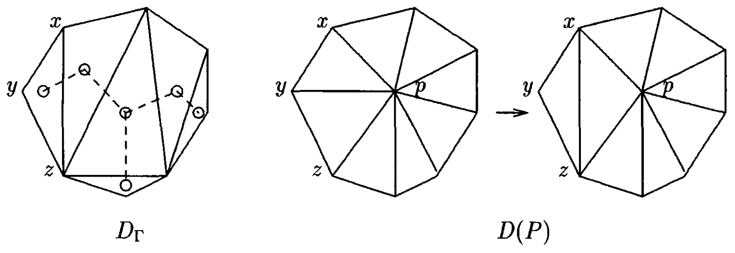


Figure 3: Leaf Pruning.

to "undo" the effects of the insertion algorithm we locate a leaf triangle $\triangle xyz$ of the D_Γ and remove this triangle *first*. Let us assume that the vertices of $\triangle xyz$ are given so that x and z are neighbors of y along the boundary of Γ). Assuming that $\triangle xyz$ does not contain p, swap the edges $\overline{xz}$ and $\overline{py}$ in the triangulation T (see Fig. 3). As a consequence, the vertex y is no longer adjacent to p. We can eliminate y from Γ, by connecting x to z, and apply the algorithm iteratively to the remaining polygon.

The complete deletion algorithm is given below. The argument p is the point to be deleted, and D is the existing triangulation. Leaf pruning is performed recursively, to emphasize its symmetry with the insertion algorithm. (Although, as in Guibas, Knuth, and Sharir [4], there does exist a purely iterative solution.)

```
**procedure** Delete($p, D$);
**begin**
    Find the set of triangles $T \subseteq D$ incident to $p$;
    Let $\Gamma$ be the polyon defined by the union of $T$;
    Compute $D_\Gamma$, the Delaunay triangulation of $\Gamma$;
    Let $\triangle abc$ be the triangle of $D_\Gamma$ which contains $p$;
    UnSwap($\overline{ca}, D_\Gamma, D$);
    UnSwap($\overline{bc}, D_\Gamma, D$);
    UnSwap($\overline{ab}, D_\Gamma, D$);
    Replace the three triangles $\triangle pab$, $\triangle pbc$, and $\triangle pca$ by $\triangle abc$ in $D$;
    Delete the triangulation $D_\Gamma$;
**end**;

**procedure** UnSwap($\overline{xz}, D_\Gamma, D$);
**begin**
    **if** $\overline{xz}$ is an exterior edge of $D_\Gamma$ **then return**;
    Let $y$ be the third vertex of the triangle to the right of $\overline{xz}$;
    UnSwap($\overline{yz}, D_\Gamma, D$);
    UnSwap($\overline{xy}, D_\Gamma, D$);
    Replace triangles $\triangle pyz$, $\triangle pxy$ with $\triangle xyz$, $\triangle pxz$ in $D$;
**end**;
```

REMARK: The most efficient way to construct the Delaunay triangulation of Γ theoretically is by the rather sophisticated linear time algorithm by Aggarwal, Guibas, Shaxe, and Shor [1]. (Although this algorithm is designed for computing the Delaunay triangulation of a convex polygon, it is shown in [1] that it can be applied to patch up a Delaunay triangulation when a point is deleted.) A much simpler but theoretically less efficient way to compute this Delaunay triangulation is to apply the incremental insertion algorithm of the previous section to the vertices of Γ given in random order. It is quite easy to show that the boundary of Γ remains intact in this triangulation (because each edge of Γ was Delaunay prior to the deletion of p), thus the triangulation of the interior of Γ can be determined by discarding all triangles which lie outside of the polygon.

This expected case $O(n \log n)$ algorithm is theoretically slower than the linear time algorithm. However, since the expected degree of a vertex in a planar graph is less than six (by Euler's formula), practically speaking the minute loss of asymptotic running time is more than compensated for by the lower constant of proportionality of the simple incremental algorithm together with the significant savings of programming effort.

Our practical experience has shown that for many natural point distributions, the maximum degree in a Delaunay triangulation rarely exceeds 16, independent of n. Thus, it may not be entirely unreasonable to apply an $O(n^2)$ triangulation algorithm. We decided against this approach because (1) the incremental algorithm is already available for our use, and (2) if there is even one vertex of degree $\Omega(n)$ in the triangulation, then the expected running time of the deletion algorithm would grow to $O(n)$. This is considerably worse than the $O(\log n)$ expected case bound, which we show below.

Given the structure of the deletion algorithm, it is relatively easy to see that it creates triangles in exactly the reverse order of the insertion algorithm, namely in a clockwise postorder traversal of the dual tree of D_Γ. If the points are in general position, and the choice of the orientation of the triangle $\triangle abc$ in procedure Delete is chosen to be identical to the triangle $\triangle abc$ of procedure Insert, then the deletion algorithm effectively swaps edges in the reverse order as the insertion algorithm (assuming that the point deleted is the last point inserted). If the points are not in general position (in particular, if four or more points are cocircular) then there may be multiple final Delaunay triangulations for Γ, thus we cannot guarantee that sequence of edge swaps is the same. We can force the orientation of the triangle $\triangle abc$ to be identical in both cases by selecting the point a in some canonical manner (e.g. by taking the point whose coordinates are lexicographically maximal).

Observe that the deletion algorithm does not need to deal with point location (except perhaps at the level of the user-interface in order to determine which point is to be deleted). However, if subsequent insertions are to be performed, the point location structures described in the previous section must be updated to reflect the change in the triangulation.

When each edge swap is performed we handle it in exactly the same way that we handled an edge swap in the case of insertion. For point bucketing the points contained within the affected triangles are redistributed among the new triangles. For the history graph the affected triangles are marked as "old" and pointers are added to the new overlapping triangles. The final step of the deletion algorithm, in which the triangles $\triangle pab$, $\triangle pbc$, and $\triangle pca$ are replaced by $\triangle abc$, is the inverse of the augmentation step seen in the insertion algorithm. For the bucketing method, the three sets of buckets are merged into a common bucket for $\triangle abc$. For the history graph method, we store a single pointer from each of the three old triangles to the newly created containing triangle.

Our next task is to analyze the complexity of the deletion algorithm. This task is complicated by the fact that the analysis of the insertion algorithm was based on the assumption that only insertions are performed and that all points are eventually added to the triangulation. To appreciate the difficulties arising when insertions and deletions can be combined, consider the case in which a single point is inserted into the triangulation and then it is deleted. This process is repeated a large number of times, N. If the bucketing method of point location is used, then when a single point is inserted all the points in the base set must be rebucketed, requiring $O(n)$ time. If the history method is used then the history graph degenerates into a structure of depth $O(N)$. Thus the expected running time in either case is much worse than the desired $O(\log n)$.

In the next section we show how to deal with the question of point location in such dynamic situations. For now, we analyze the expected running time of one deletion. This running time follows almost directly from the analysis of the insertion algorithm. Because the particular edge swaps and point movements (arising from point location) for deletion are just the reverse of those for insertion, the expected number of edge swaps and point movements needed to delete a random point p from a triangulation $D(P)$, is identical to the expected number of edge swaps and point movements needed to insert the random point p into the triangulation $D(P - \{p\})$. Under our assumptions of random point insertion and deletion, the sets P and $P - \{p\}$ are random point sets. Thus, this portion of the cost of deleting a random point from a triangulation n points is $O(\log n)$ in the expected case.

The only other aspect of the complexity of deleting a point p is the cost of computing the Delaunay triangulation of Γ, the polygon of neighboring vertices. The number of vertices in Γ is equal to the degree of p in the triangulation. To establish the expected cost of this operation, let $d_1, d_2, \ldots, d_n$ denote the degrees of each of the n vertices of the triangulation. By Euler's formula we know that the sum of degrees, which equals twice the number of edges in the graph, is at most $6n$. By the analysis of the previous section, the expected time to compute the Delaunay triangulation of a set of d_i points is $O(d_i \log d_i)$. Thus, the expected time needed to compute one such Delaunay triangulation, under the assumption that each point is equally likely to be deleted is

$$\frac{1}{n}\sum_{i=1}^{n}(d_i \log d_i) \leq \frac{1}{n}\left(\sum_{i=1}^{n} d_i\right)\log n \leq \frac{6n}{n}\log n = 6\log n.$$

Thus, the expected time to delete a point from the triangulation is $O(\log n)$, from which we have our main result.

THEOREM 3.1 *Given the above deletion algorithm (ignoring point location issues) a point can be deleted from a Delaunay triangulation of n points in expected $O(\log n)$ time, under the assumption that each point of the triangulation is equally likely to be deleted.*

4 Sequences of Insertions and Deletions

As we mentioned in the previous section, in the worst case, where long sequences of insertions and deletions are made, the expected case running time of the algorithm can be much larger than $O(\log n)$. In this section we consider how to deal with the problem.

Our first observation is that in certain relatively benign cases (which may be quite common in many practical applications) there is really no problem at all. For example, if insertions are more common than deletions (in the sense that the ratio of the number of insertions to deletions is strictly greater than unity) then it follows that over a long sequence, the cost of updating the triangulation is dominated by the costs of the insertions. Although the deletions cause an increase in the size of the history graph, the assumption of randomness implies that these variations in the history graph are distributed evenly throughout the graph, and it was shown in the previous section that the local effect of each deletion on the structure is essentially equivalent to the effect of an insertion.

In steady state situations, where the number of active points in the triangulation reaches an equilibrium, a direct application of the deletion and insertion algorithms is rather unpredictable. If the number of active points is a roughly a constant fraction of the total number of base points, then the randomness of insertion and deletion, combined with Guibas, Knuth, and Sharir's arguments about the widths of triangles, imply that only a constant number of points will be rebucketed with each change to the triangulation. However, if the number of active points is significantly less than the total number of base points, then nearly all of the nonactive base points may be rebucketed with each update. The history method will fair even worse, because irrespective of the number of active points, the history graph grows without bound as updates are made.

In this section we will consider how to periodically rebalance the history graph so that these problems can be avoided. The idea is that from time to time, we will completely reconstruct the history graph from scratch for only the current set of active points, and destroy the old graph. (Observe that this will have the unfortunate consequence of destroying the persistence property provided by the unpruned structure.) We refer to this process as *reorganization*. Let n denote the number of active (triangulation) points. We show that by applying reorganization at appropriate times, we can maintain an $O(\log n)$ expected time cost for insertion or deletion, when amortized over sequences of insertions and deletions. The expectation here is over possible random choices of which point to insert or delete. The choice of whether to insert or to delete is arbitrary.

We assume initially that the triangulation is trivial (consisting only of the four vertices and two triangles of the unit square). Let t denote the total number of insertion/deletion requests which have been performed, and let $n(t)$ denote the number of active points in the triangulation after the t-th request. Thus, $n(0) = 4$. Let t_0 be the time (i.e. the request number) of the last reorganization. After performing the t-th operation we test whether

$$t - t_0 > n(t).$$

If this is the case then reorganization is performed. Reorganization consists of first discarding the existing history graph and triangulation, and then constructing a new history graph and Delaunay triangulation by inserting (in random order) each of the current active points.

THEOREM 4.1 *Using this reorganization scheme, the expected amortized time for processing an insertion or deletion request for a random point is $O(\log n)$, where n is the number of active points at the time of the insertion or deletion.*

PROOF: The theorem follows from two observations. The first is that if periodic reorganization is applied, then the execution time of insertion or deletion at any time (ignoring reorganization) is $O(\log n)$. The second observation is that reorganization is performed infrequently enough that it does not increase the asymptotic running time of the algorithm.

To prove the first observation, let n_0 denote the number of active points at the time of the most recent reorganization. Because this is the first point at which we have performed a reorganization since t_0, it follows that for all s, $t_0 \leq s < t$, $n(s) \geq s - t_0$.

The number of nodes in the history graph increases by an expected constant amount with each insertion or deletion (since the expected number of new edge swaps is constant). Thus the expected size of the history graph after the s-th request is roughly proportional to $n_0 + (s - t_0)$.

Since we can lose at most one point at each insertion/deletion request, we have $n(s) + (s - t_0) \geq n_0$. Thus

$$n_0 + (s - t_0) \leq n(s) + 2(s - t_0) \leq n(s) + 2n(s) = 3n(s).$$

In other words, at no time is the expected size of the history graph significantly larger than the number of active points. Because changes to the history graph are made randomly throughout its structure, it follows that the cost of searching the graph does not increase asymptotically, so the search time is $O(\log(n_0 + s - t_0)) = O(\log n(s))$. All other aspects of the insertion and deletion routines are $O(\log n(s))$ running time. This establishes the first observation.

To establish the second observation, observe that the expected time to perform reorganization is $O(n \log n)$, where n is the number of active points. Since the expected case of insertion or deletion ignoring reorganization is $O(\log n)$, it follows that the cost of reorganization will not dominate the overall cost if the number of insertions or deletions since the last reorganization is at least as large as $n = n(t)$. However, in order to perform reorganization it must be the $t - t_0 > n$, thus the number of requests which have been processed since the last reorganization is at least as large as the number of active points. This establishes the second observation. □

5 Implementation Experience

In this section we discuss our implementation of the algorithm. The algorithm has been implemented in the C programming language, under the Unix operating system. (Currently the reorganization scheme has not been implemented.) It has been designed to provide statistics on the execution of the algorithm for the purposes of evaluating its efficiency. Rather than measuring execution time by CPU seconds, because of its dependence on the particular machine and compiler, we have measured two quantities which we feel give a strong indication of the algorithm's general performance. First, we have measured the number of times that a point moves from one triangle to another in the bucketing algorithm (equivalently, the number of levels that each point travels through the history graph), and second we have measured the number of edge swaps which were performed.

We have run the following experiments involving point insertion. (The reasons that we did not consider deletion are (1) the number of edge swaps and point movements for deletion are identical in the expected case to insertion, and (2) we have not yet implemented the reorganization scheme described in the previous section.)

Uniform Data: Points were sampled from a uniform distribution over the unit square and inserted into the triangulation. Point sets of size 50, 100, 200, 400, 800, 1600, 3200, and 6400 were considered.

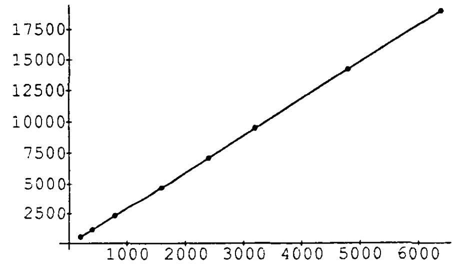
Gaussian Data: Points were sampled from a Gaussian distribution whose center is at the center of the unit square and whose standard deviation was 0.2 in each of the x and y directions. Point sets of the same sizes as in the uniform case were considered.

Sorted Data: To test the sensitivity of the algorithm to violations of the randomness assumption, we ran an experiment in which points were selected uniformly from the unit square, but were inserted in order of increasing x-coordinate.

Partially Random Data: In this variant of the previous experiment, we inserted points in which the first p points were inserted randomly (out of a total of 6400), and the remaining points were inserted in order of increasing x-coordinate. The values of p tested were 25, 50, 100, 200, 400, 800, 1600, 3600, and 6400.

The results of the these experiments are given below.

Uniform Data: Fig. 4(a) shows a plot of the number of edge swaps performed by the algorithm versus n, the number of points. The regression line fitted to the data is $2.97n - 68.0$. Fig. 4(b) shows a plot of $\log_{10} n$ versus the average number of point movements per point. Standard deviations are indicated by vertical lines. The regression line fitted to the data is $9.02 \log_{10} n - 3.75$.



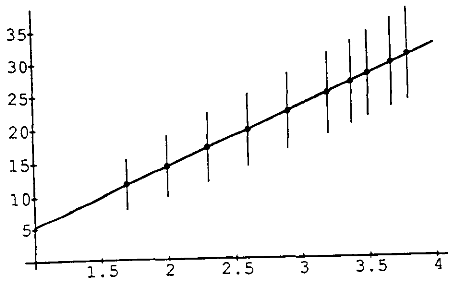
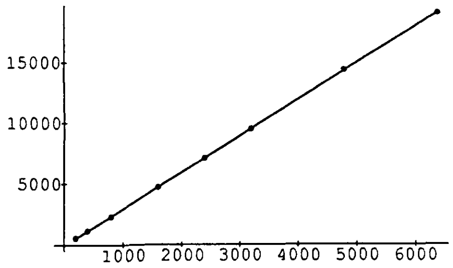


Figure 4: Uniform data: Total edge swaps and average point moves.

Gaussian Data: Analogous results for Gaussian data are shown in Figs. 5(a) and (b). In the first case the regression equation is $3.01n - 77.4$, and in the second case it is $9.66 \log_{10} n - 6.32$. Both cases are in close agreement with the uniform case, although the number of edge swaps is slightly larger in the Gaussian case.

Sorted Data: Fig. 6(a) shows a plot of the total number of edge swaps performed versus n in the case that points are inserted in sorted order. The regression line fitted to the data is $4.72n - 494$. The slope is greater than the previous cases, however this supports the observation made by Tipper [9] that the average number of edge swaps per point is independent of n. However, the plot of $\log_{10} n$ versus the number of point moves showed a striking nonlinear behavior (see Fig. 6(b)). It is clear that the assumption of randomness is critical to the analysis of the point location schemes.



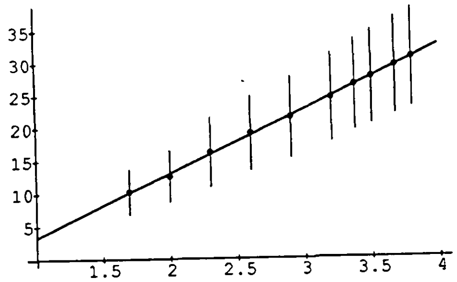
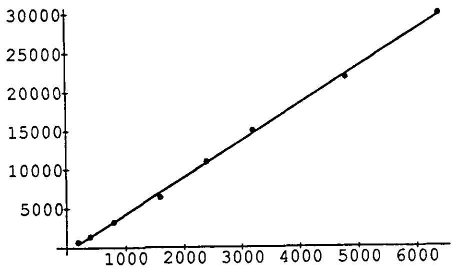


Figure 5: Gaussian data: Total edge swaps and average point moves.



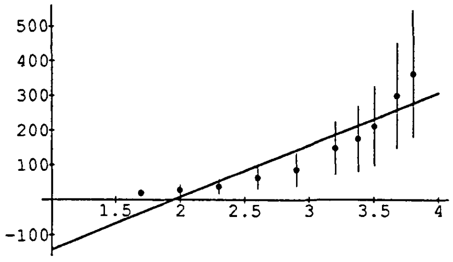


Figure 6: Sorted data: Total edge swaps and average point moves.

Partially Random Data: The results of the previous experiment lead us to the question of how many initial points need be inserted randomly in order to guarantee fairly good performance in point location. Fig. 7 shows $\log_{10} p$ versus the average number of movements per point. Interestingly, with as few as 200 of the 6400 points inserted randomly (about .03% of the total) the performance is within a factor of 2 of the totally random case.

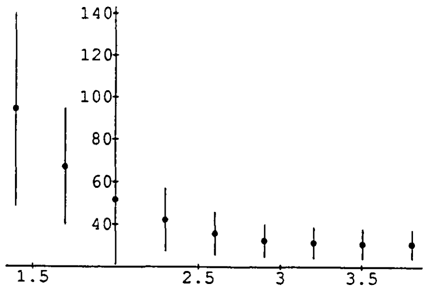


Figure 7: Partially random data: Average point moves.

6 Conclusions

We have presented and analyzed the complexity of a procedure for computing and updating Delaunay triangulations for point insertion and deletion. The algorithm is randomized and incremental, based on a recent algorithm of Guibas, Knuth, and Sharir. The algorithm has the nice feature that it is asymptotically as efficient (in the expected case) and yet much simpler than standard divide-and-conquer algorithms. Its expected running time is independent of the distribution of the points, only on the order in which the points are inserted or deleted. We have implemented a portion of the algorithm for the purpose of empirical analysis. Our studies seem to indicate that the running time is quite good even if the assumption of random insertion order is violated as long as an initial fraction of the points are inserted randomly.

One interesting open problem raised by this research is whether these results can be applied to more general types of triangulations. In particular, in geographic information systems, it is quite common to require that certain edges be present in the Delaunay triangulation, giving rise to a *constrained Delaunay triangulation.* It would be of interest to develop and analyze the performance of a randomized incremental algorithm for constrained triangulations.

References

1. A. Aggarwal, L. J. Guibas, J. Saxe, P. W. Shor, A linear-time algorithm for computing the Voronoi diagram of a convex polygon, *Discrete and Computational Geometry* 4 (1989), 591–604.

2. S. Fortune, A sweepline algorithm for Voronoi diagrams, *Algorithmica*, 2 (1987), 153–174.

3. P. Green and R. Sibson, Computing Dirichlet tesselation in the plane, *Comput. J.* 21 (1977), 168–173.

4. L. Guibas, D. Knuth, and M. Sharir, Randomized incremental construction of Delaunay and Voronoi diagrams, Unpublished manuscript (1990), also appeared in the *Proceedings of ICALP*, 1990.

5. L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Trans. on Graphics*, 4 (1985), 74–123.

6. M. Heller, Triangulation algorithms for adaptive terrain modelling," *4th Symposium on Spatial Data Handling*, 1990, 163–174.

7. M. Mäntyllä, *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland, 1988.

8. T. K. Peucker, R. J. Fowler, J. J. Little, and D. D. Mark, Digital representation of three dimensional surfaces by triangulated irregular networks. Tech. Report #10, ONR Contract N00014-75-C-0886, 1976.

9. J. C. Tipper, A straightforward iterative algorithm for the planar Voronoi diagram, *Information Proc. Letters* 34 (1990), 155–160.

ADAPTIVE HIERARCHICAL TRIANGULATION†

Lori Scarlatos
Grumman Data Systems
1000 Woodbury Rd.
Woodbury, NY 11797

Theo Pavlidis
State University of New York
Dept. of Computer Science
Stony Brook, NY 11794

ABSTRACT

Numerous cartographic applications rely on triangulated surface models for accurate three-dimensional representations of real-world data. Some applications require a series of triangulations to represent a single surface at progressively finer levels of detail. Past work has emphasized techniques relying on plane geometry using little or no surface data. We propose a technique that adapts the triangulation to surface characteristics. Because our adaptive hierarchical triangulation focuses on the topology of a surface, it reduces the number of triangles required for a good approximation. It also produces fewer long and slivery triangles within each level of detail. Our structure guarantees the accuracy of each level of detail. Our structure only retains important triangles, thereby reducing the total number of triangles that must be stored and searched. Furthermore, the tree-like structure of our hierarchy is well-adapted to multiple resolution views, allowing smooth transitions between levels of detail in flight simulators. These advantages add up to a triangulation that provides great accuracy in a model that can be rapidly searched, rendered, and otherwise manipulated. Tests on data with digital elevation input have confirmed the above theoretical expectations. On eight such sets the average ''sliveriness'' with the new method was between 1/5 and 1/10 of old triangulations and number of levels was about one third. Although the number of descendants at each level increases slightly, the total number of triangles is lower, implying faster spatial search.

INTRODUCTION

Geographic information systems, flight simulators, and numerous other cartographic applications rely on digital terrain models for simulation, visualization, and analysis. Increasingly, these applications require both greater accuracy and data compression from these models. Triangulated models are popular because triangles are simple to manipulate and render. Triangulated Irregular Networks (TINs) offer the additional

† Portions of this paper were included in a paper presented at Visualization '90. For further details and examples of this work, interested readers should request a technical report entitled "Hierarchical Triangulation Using Terrain Features" from Lori Scarlatos.

advantage of not being bound by regularity constraints. TINs can therefore approximate any surface at any desired level of accuracy using a very small number of polygons. Organizing TINs in a level of detail hierarchy provides accurate generalizations meeting different application requirements. Hierarchical organization allows easy implementation of such operations as zooming when viewing the surface. It also facilitates searching and other geometrical operations such as finding the intersection of two surfaces. Furthermore, it makes real-time simulation and visualization possible for applications that can represent less important areas with less detail in mixed-resolution models.

This paper describes a hierarchical triangulation built from a digital elevation model in grid form. Each level in the hierarchy corresponds to a different level of detail that approximates the surface within a given tolerance (i.e. maximum error), The top level is the coarsest, containing the fewest triangles and approximating the surface within the greatest tolerance value t_0. The $i+1^{th}$ level in the hierarchy is related to the i^{th} level as follows. Tolerance t_{i+1} is smaller than t_i. Each triangle T_j^i of the i^{th} level is split into n descendent triangles $T_{j_1}^{i+1}, \cdots, T_{j_n}^{i+1}$ at the $i+1^{th}$ level, where n can be any positive integer.

In the following section, we provide a background of past work on triangulation and hierarchical triangulation. We then describe our adaptive hierarchical triangulation methodology, and discuss its advantages over other methods. Next, we outline the implementation of this algorithm and the resulting data structure. We conclude with a discussion of test results obtained from running this implementation.

BACKGROUND

Triangulation

Triangulation algorithms generally fall into two groups: those that efficiently triangulate a given polygon, and those that use triangulation to approximate surfaces. In the former category, the primary issues are computational complexity (Aho et al 1974, Garey et al 1978, Clarkson et al 1989, Fournier and Montuno 1984) or size and shape of the resulting triangles (Baker et al 1988). We are more interested in the latter category where the primary goal is to produce the best possible surface approximation. This surface approximation should contain as few triangles as possible while still meeting given accuracy requirements. At the same time, it must minimize the number of very thin, slivery triangles which can produce artifacts in renderings of surface models.

Surface triangulation algorithms may be further categorized by the input data they triangulate. Surface triangulation produces a planar graph by adding edges, and sometimes even points, to an initial graph. This initial graph, comprised of points (nodes) on the surface, may or may not include connecting edges (critical lines) that further define that surface.

In the first sub-category of surface triangulation algorithms, the initial graph contains no initial edges. Although some of these triangulation algorithms rely on alternate techniques (Mirante and Weingarten 1982, Manacher and Zobrist 1979) most are a variation on the Delaunay triangulation scheme (Watson 1981, Dwyer 1987, Preparata and Shamos 1985, Watson and Philip 1984 are only a few). Algorithms based on Delaunay triangulation have the advantage of producing few slivers. However, Delaunay's method was developed to find nearest neighbors on a plane, not approximate surfaces. These algorithms tend to ignore the third dimension, and may therefore produce triangle edges that contradict the topology of the actual surface (Christensen 1987).

The second sub-category of algorithms assumes that all points in the initial graph have at least one connecting edge. These edges correspond to the linear patterns that characterize many surfaces, particularly natural ones such as terrain. Because these edges describe surface topology, they are retained in the final triangulation to maximize model accuracy. Some papers such as (Christiansen and Sederberg 1978, Dennehy and Ganapathy 1982) deal with triangulating cross-sections from tomographic scans, although the methods of both of these papers require human intervention when the contours get complex. Other algorithms for triangulating cartographic critical lines have been recently published (Christensen 1987, Scarlatos 1989, Chew 1989).

Hierarchical Triangulation

Hierarchical triangulations provide both multiple levels of detail and a structural ordering for fast spatial search. Recent papers (Goodchild 1989, Fekete 1990) propose to represent the entire planetary surface with a quadtree-like hierarchy of regular triangular tessellations. This is an excellent scheme for dividing huge data bases into manageable areas of interest which may be georeferenced in constant time. However, as shown in (Scarlatos 1990b), the placement of points in a regular tessellation is independent of the surface topology. Hence coarser levels of detail can distort or entirely miss important terrain features, and finer levels of detail can cause unnecessary bottlenecks by producing large numbers of triangles where a few would do as well.

Previous work by one of the authors has researched techniques to find critical points and lines (Scarlatos 1990a), triangulate them (Scarlatos 1989) and then refine those triangulations to produce a hierarchy of detail levels for fast spatial search with maximum accuracy (Scarlatos 1990b). These algorithms represent significant improvements over other algorithms, producing good triangulations. However, the above algorithms do not allow for refinement down to a specified level of accuracy.

Although several refinement techniques have been suggested in the literature (Fowler and Little 1979, DeFloriani et al 1984, DeFloriani 1989), these algorithms can introduce artifacts to a terrain model because they consider only the locality of points in a 2D plane instead of actual terrain topology. Consider, for example, DeFloriani's first algorithm for triangle refinement (DeFloriani et al 1984) which splits a triangle by connecting its corners to a selected interior point (usually, the point of maximum distance between the surface and the plane defined by the vertices of the triangle). The algorithm ignores the *coherence* of cartographic features such as valleys or ridges which have a linear structure.

Figure 1 shows the results of ignoring such coherence. We assume that a ridge (its points marked by small circles in (a)) crosses the triangle. (b) shows that the maximum point triangulation will produce an unreasonably large number of triangles. Even worse, the triangles will have very sharp angles, which is an undesirable property (Baker et al 1988). Such triangulations may cause numerical stability problems in finite element methods and also produce undesirable display artifacts. In contrast, if we realize that we deal with a ridge and introduce a dividing line along it as shown in (c) we will end up with fewer triangles, none of them slivery. We should point out that triangles with very sharp angles may be inevitable for some types of data. For example, if we have a steep cliff we will see large differences in the value between adjacent elevation points. Then triangles with very sharp angles cannot be avoided.

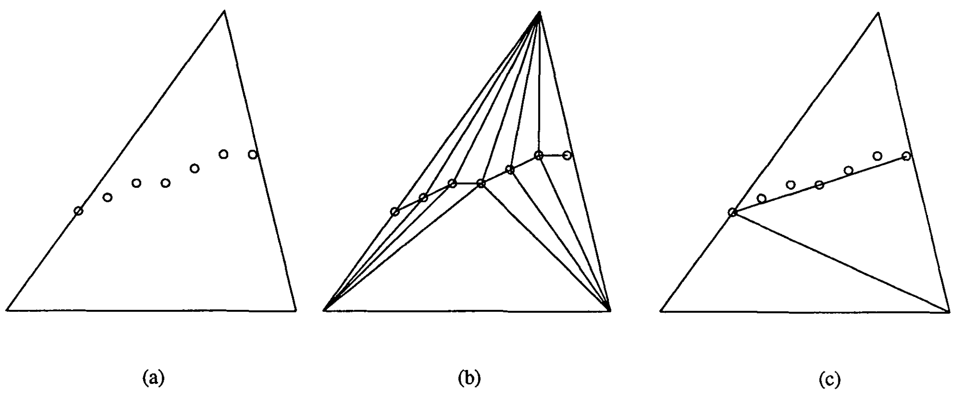


Figure 1. A ridge passing through a triangle has (a) points along that ridge that are farthest from the triangle which may be triangulated using (b) the maximum error point to split each triangle, or (c) cartographic coherence to approximate the ridge line.

METHODOLOGY

Our goal is to reduce the number of splits or refinements required to achieve a desired level of detail and limit the number of slivery triangles in the results. A generalization of the critical line method could produce better accuracy with fewer triangles. We have implemented such a strategy as follows. We start with a coarse triangulation. This may be carefully produced by techniques outlined in Scarlatos' three papers, or it may be as simple as a rectangular area split in two. We then refine this triangulation by adding points from the original digital elevation grid and connecting edges. Our refinement technique pays particular attention to terrain characteristics, approximating critical lines at each step.

To accomplish this, we determine the best places to split each triangle by calculating four error values: one inside the triangle, and one on each of the three edges. All error values measure the difference between original grid point elevations and their projections to the surface of the triangulated model. To avoid quantization artifacts, grid points near a triangle edge are considered to be on that edge.

Figure 2 shows the five ways that a triangle may be refined. If an isolated peak or pit resides within the triangle, it is split at that central peak or pit point as shown. If a single ridge or channel line travels up to that peak or pit, the triangle is split where that line crosses the edge of the triangle and at the central peak or pit point. If, however, a single ridge or channel line enters the triangle and ends at a saddle point or flat, then the center point is insignificant and the triangle is split by one edge as shown. If a ridge or channel line passes through the triangle, significant errors will be found on two edges of the triangle. A line connecting these points approximates the topographical line, and an additional edge splits the remaining quadrilateral. Finally, if a triangular patch corresponds to a rapidly fluctuating surface, many points are likely to have significant errors. Splitting this type of triangle on all edges segments the high-frequency regions which may then be further refined.

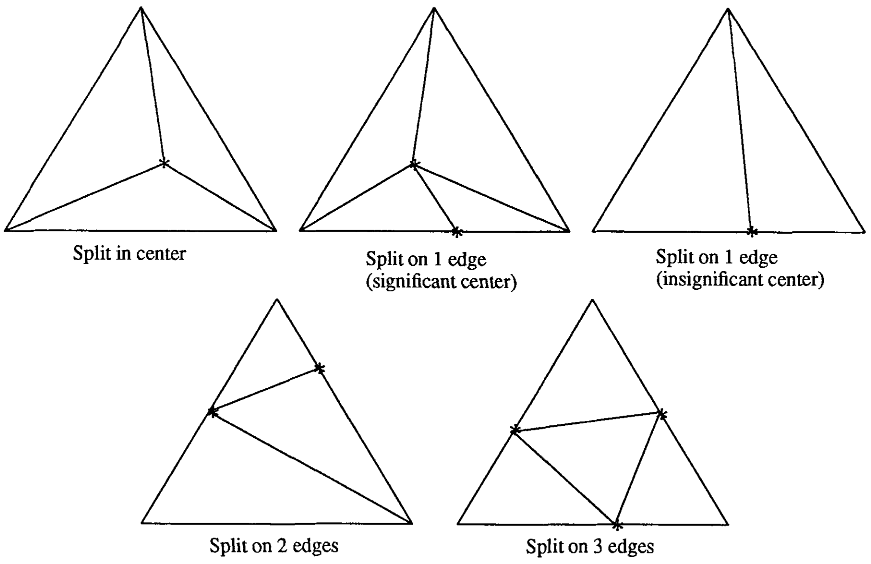

Split in center
Split on 1 edge (significant center)
Split on 1 edge (insignificant center)
Split on 2 edges
Split on 3 edges

Figure 2. Split strategies for preserving cartographic coherence.

We repeatedly split the triangles until they all meet the given accuracy requirements for the current level of detail. Intermediate triangles, used to produce but not included in the final triangulation for the current level of detail, are discarded. This reduces the number of levels in the hierarchy and the number of triangles within each level, making faster search, display, and processing possible. If polygon constraints are more important than the level of error, we can easily check the polygon count and terminate a level when the limit is approached.

IMPLEMENTATION OF THE ALGORITHM

We implemented our adaptive hierarchical triangulation algorithm as follows. A main program retrieves the input data, calls the appropriate triangulation routines, and writes out the results to a data base. Input parameters include an initial triangulation, the number of levels to create in the hierarchy, and a tolerance for each level. A main loop generates each level of detail. At the start of the loop, the current triangulation represents level i in the hierarchy. At the conclusion of the loop, the current triangulation represents level $i+1$ in the hierarchy. The body of the loop splits triangles in the current triangulation until all errors lie within the given tolerance for that level.

Data Structures

We generate our adaptive hierarchical triangulation from a digital elevation matrix which covers a rectangular area of interest. The region outside the area of interest is represented by four neighboring "triangles". These extend infinitely to the north, east, south, and west of the area of interest. Points within the area of interest provide the endpoints of -- and are entirely covered by -- an initial triangulation. Each point may therefore be associated with zero, one, or two triangles. Points acting as triangle vertices have no triangle associations. If the distance from a point to a triangle edge is less than the distance between grid posts in the original matrix, then that point is considered near that edge. A point on or near a triangle edge is associated with the two triangles that share that edge. Otherwise, the point is within a single triangle. A Membership list contains records of each point's two associated triangles and a distance to their shared edge. When a point is near more than one edge, the membership records form a linked list in order of increasing distance values.

A triangle in the hierarchy is defined by three points from the original elevation matrix. Each triangle is associated with a level of detail and contains pointers to its parent, its children, and three neighboring triangles that share its edges. In addition, triangles have temporary structures keeping track of their splitting points, the maximum error found within them, and the number of edges to be split. A flag indicates whether the triangle meets the accuracy standards of the current level.

Splitting Triangles

For each specified level of detail, our program repeatedly splits triangles until the triangular mesh approximates the surface within the given tolerance. We find errors within a triangle by taking all grid points within the boundaries of that triangle, projecting them to the surface of the triangle, and comparing the results to the original elevation values. Errors are found in four regions on a triangle: on each of the three edges, and within the triangle. These errors determine if and how the triangle will be split.

Next, we find the point producing the maximum error in each of the four regions for each triangle. Notice that the point with maximum error on one triangle's edge will also be the point with maximum error for the other triangle sharing that edge. If the error is significant, then that point will split the triangle(s) it belongs to. Significance may be calculated in two ways. First, if the given value is greater than the threshholded error for the current level of detail, then that point is significant. Alternatively, if the given value is more than some percentage of the maximum error found within a triangle, then that point is significant. In either case, a point is insignificant if its error falls at or below the threshhold for the current level of detail.

After all splitting points have been found, we ensure that the splitting point on an edge of one triangle is also a splitting point for the triangle sharing that edge. Then we split all the triangles. Although each of the five regular triangulation algorithms is different, they all follow the same pattern of steps. First the outer edges of the triangle are split. If the splitting point does not lie exactly on the outer edge, this will introduce a minor bend in the triangle. Extremely thin triangles produce special cases which must be handled separately. Our technical report discusses the necessary special triangulation in depth. In the next step we add all the new interior edges. As we modify and add edges, we update the point membership list indicating what triangle(s) each point belongs to. Finally, new triangle records are added, and triangle neighbor values are updated.

Data Base Structure

This algorithm produces all of the information required to both render the 3D surface and search for spatial relationships. A header record includes information such as a ground position for the lower-left corner of the triangulation; spacing between posts in the original grid; elevation ranges in the triangulation; number of levels. This is followed by the level records. Each level has a threshhold of allowable error, used to produce the triangulation. It also has a number of points, number of triangles, and a list of triangles. Each triangle is defined by 3 point indices, and has a parent pointer, child pointers, and neighbor pointers. All this is followed by a single point list. Only points that appear in the triangulation are written to the data base; all others are unnecessary. Points are ordered such that if level L uses N points, then it uses points $1, \cdots, N$. This reduces retrieval time for a level of detail.

RESULTS

We tested our algorithm on eight (8) areas of interest (AOI) representing four very different types of terrain. AOI 1-2 contains numerous plateas; AOI 3-4 is a relatively flat region; AOI 5-6 contains mountains rising out of the foothills; and AOI 7-8 represents a portion of the Cascade mountain range. Our data comes from the Defense Mapping Agency's Digital Terrain Elevation Data (DTED) Level 1 which has three seconds of an arc between posts. Each test AOI covers 75x75 elevation points. A triangulation employing all 5625 points in an AOI would contain 10,952 triangles.

We implemented the adaptive hierarchical triangulation algorithm with varying parameters to see which behaved best. The first parameter is how we determine the significance of point p's error e_p. Error e_p may be considered significant compared to 1) tolerance value t_i for level i, so that $e_p \geq t_i$, or 2) a percentage N of the maximum error $e_{t_{\max}}$ found for current triangle t, so that $e_p \geq e_{t_{\max}}$. The second parameter determines when we split a triangle at one edge and a significant center (as shown in Figure 2). Center point c may be considered significant compared to 1) the error e_v of splitting point v on the edge of the triangle, so that $e_c \geq e_v$, or 2) the significance value used to determine the significance of all other points, as determined by the first parameter. Hence we ran four optional programs. Option 1 uses tolerance to determine significance, and requires a center point to be at least as significant as an edge point in order to be used. Option 2 uses 75% of the maximum error within a triangle to determine significance, and also requires a center point to be at least as significant as an edge point. Options 3 and 4 are like options 1 and 2 respectively, except that a center point's significance is determined by the usual measures. As a basis of comparison, we implemented DeFloriani's first algorithm (DeFloriani et al 1984) and ran it with the same test data.

We executed DeFloriani's algorithm and all four options for our algorithm using the eight AOIs as input, producing triangulations with a minimum error of 10 meters. All tests demonstrated that adaptive hierarchical triangulation works well. Tables 1-4 show some of our results.

A better triangulation will produce fewer slivery triangles. The table shows how slivery the resulting triangles were. We measured sliveriness with the following ratio, calculated for each triangle in the triangulation: $\frac{Perimeter^2}{Area}$. The best possible ratio is approximately 20.78 for an equilateral triangle. Larger values represent thinner triangles, so smaller numbers are better. We summed all of these ratios together and divided by the total number of triangles to get an average sliveriness figure. We then divided that result by the sliveriness ratio for an equilateral triangle. Note that on the average most of the triangles have much sharper angles than sixty degrees. Using DeFloriani's algorithm, some angles are as small as 0.25 degrees. Notice how much better adaptive hierarchical triangulation performed, using all four options. Options 1 and 2 seem to work about equally well, indicating that the best measure of point significance is determined by data characteristics. Options 3 and 4 consistently performed a little worse. This leads us to conclude that a center point should only be included in the division of a triangle if it is more significant than the edge point.

Table 1

Measures of Sliveriness*					
AOI	DeFloriani-1	Option 1	Option 2	Option 3	Option 4
1	32.294	5.086	5.113	6.301	6.578
2	52.487	9.645	11.882	11.074	11.107
3	35.889	5.934	5.672	6.398	5.998
4	50.682	11.315	10.969	12.581	12.854
5	56.835	14.843	8.757	14.329	8.676
6	40.932	5.305	5.371	7.376	7.437
7	51.261	4.352	4.932	6.089	7.367
8	39.925	5.949	6.200	6.805	7.153

* normalized to 1 for an equilateral triangle

Table 2

Comparison of Hierarchies				
AOI	Number of Levels*		Average Number of Children**	
	DeFloriani-1	Option 1	DeFloriani-1	Option 1
1	15	5	2.5	2.8
2	17	5	2.5	2.4
3	17	5	2.5	3.6
4	17	5	2.5	3.6
5	19	5	2.5	2.4
6	18	5	2.5	2.5
7	17	5	2.5	2.4
8	18	5	2.5	2.3

* number of levels specified for new algorithm

** number of children assumed to be 2.5 for old algorithm

A better triangulation will permit fast spatial search. The time required for a search is determined by the number of levels that must be searched, and the number of child nodes that must be examined at each level. DeFloriani's algorithm, which always splits a parent triangle into 2 or 3 children, has an average of about 2.5 children per parent node. The number of levels in a hierarchy depend on the number of iteration levels required to build the triangulation. Adaptive hierarchical triangulation, on the other hand, guarantees a fixed number of levels in the hierarchy, but can split a parent triangle into any number of children. Although one may presume that a very large number of children will be produced, table 2 shows that this is not the case. Table 2 shows that search times using an adaptive hierarchical triangulation will be as fast as, or faster than, the other. Additional results can be found in our technical report.

A better triangulation will result in fewer triangles. Table 3 shows the total number of triangles in the hierarchy. Notice that the options that produced the fewest total triangles also produced the least slivery triangles. Table 4 shows the number of triangles at the highest level of detail, with a maximum error of 10 meters at each point. Compare this to 10,952 triangles for the original grid. Although the difference in values is not striking, the adaptive hierarchical triangulation usually produced fewer triangles than DeFloriani's algorithm.

Figure 3 demonstrates the significance of the improvements made by the adaptive hierarchical triangulation. Figure 3 a shows a view of AOI 1 using the original grid data. Figure 3 b shows the same view of the data triangulated with DeFloriani's algorithm for a maximum error of 10 meters. Figure 3 c shows the same view of the data triangulated with our algorithm (using option 1) for a maximum error of 10 meters. All three views were rendered with Gouraud shading. Notice the severe artifacts caused by very thin triangles in the DeFloriani model.

While Delaunay triangulations have been proposed as means for reducing the number of very sharp triangles within hierarchical structures (DeFloriani 1989), Delaunay triangulations have serious drawbacks as discussed in (Christensen 1987). In some cases, using Delaunay triangulation to add points can actually increase error levels in the model, even though the model contains more triangles. The algorithm of Baker et al (1988) while it avoids generating obtuse triangles, it generates far too many points and triangles for our purposes.

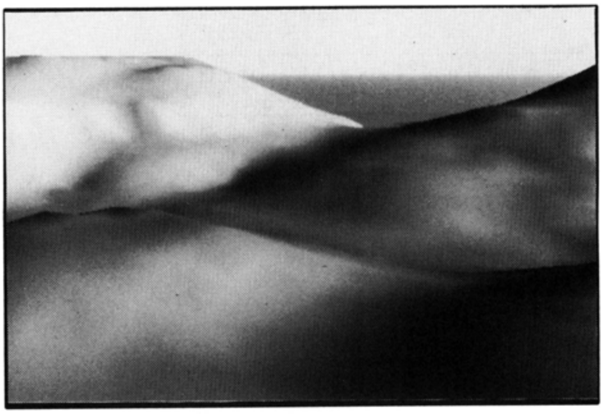
Table 3

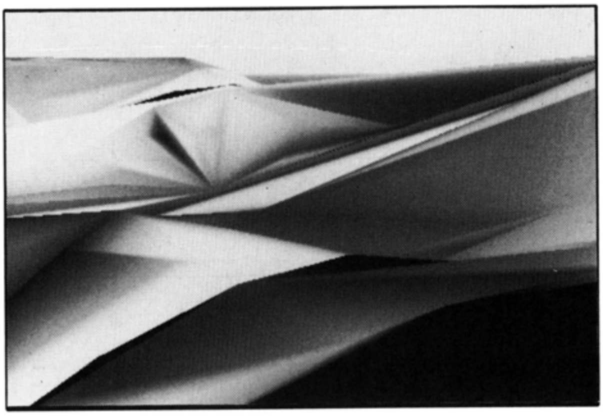
Total Number of Triangles in the Hierarchy					
AOI	DeFloriani-1	Option 1	Option 2	Option 3	Option 4
1	2918	2866	2876	3208	3195
2	4198	3964	4124	4344	4364
3	2576	1862	1806	2022	2055
4	2007	1551	1525	1737	1757
5	5433	5339	5179	5508	5372
6	3935	3283	3289	3624	3604
7	4908	4655	4735	4995	5109
8	7962	7927	8125	8312	8516

Table 4

Number of Triangles* in Highest Level of Detail (Tolerance = 10 meters)					
AOI	DeFloriani-1	Option 1	Option 2	Option 3	Option 4
1	1741	1852	1858	1942	1935
2	2474	2330	2380	2442	2452
3	1547	1353	1309	1439	1470
4	1211	1123	1111	1196	1237
5	3185	3072	3062	3127	3135
6	2318	1979	1992	2167	2137
7	2883	2745	2769	2899	2901
8	4568	4418	4414	4436	4586

* compare to 10952 triangles in grid


(a)


(b)

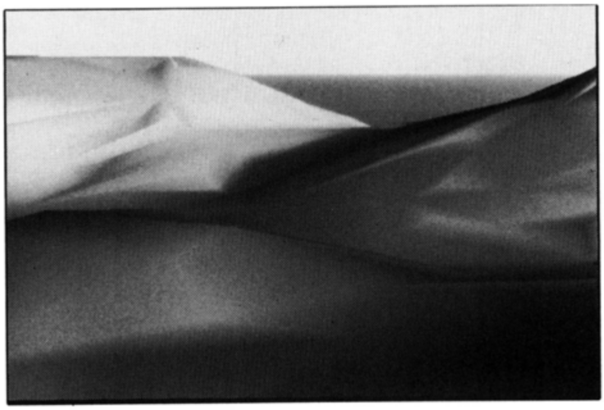

(c)

Figure 3. Perspective views of AOI 1 modeled with (a) DTED, (b) DeFloriani's algorithm, (c) Adaptive Hierarchical Triangulation.

CONCLUSIONS

Adaptive hierarchical triangulation, presented in this paper, has the following advantages over other algorithms currently used. First, because our algorithm focuses on the topology of a surface, it reduces the number of triangles required to accurately approximate the surface and produces fewer long and slivery triangles within each level of detail. Second, our structure guarantees the accuracy of each level of detail. This may be easily extended to impose a polygon limit at each level. Third, our structure only retains important triangles, thereby reducing the total number of triangles that must be stored and searched. Fourth, the tree-like structure of our hierarchy is well-adapted to multiple resolution views, allowing smooth transitions between resolutions in animation. Because adaptive hierarchical triangulation pays attention to surface topology, this transition from low to high levels of detail will cause only minor terrain features to appear. Finally, adaptive hierarchical triangulation algorithm is fully automated, requiring only the area of interest and a series of tolerance levels to be defined. This algorithm can also be shown to run in $O(n \ln n)$ time. These advantages add up to a triangulation that provides great accuracy in a model that can be rapidly searched, rendered, and otherwise manipulated.

REFERENCES

Aho, A.V., Hopcroft, J.E. and Ullman, J.D., 1974. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass.

B. S. Baker, E. Grosse and C. S. Raferty, 1988. Nonobtuse triangulation of polygons, *Discrete Computational Geometry*, 3, 147-168.

Chew, L.P., 1989. Constrained Delaunay triangulations, *Algorithmica*, 4, 97-108.

Christensen, A.H.J., 1987. Fitting a triangulation to contour lines, *Proceedings of AUTO-CARTO 8*, 57-67.

Christiansen, H.N. and Sederberg, T.W., 1978. Conversion of complex contour line definition into polygonal element mosaics, *Proceedings of SIGGRAPH '78*, 187-192.

Clarkson, K.L., Tarjan, R.E. and Van Wyk, C.J., 1989. A fast Las Vegas algorithm for triangulating a simple polygon, *Discrete and Computational Geometry*, 4(5), 432-432.

DeFloriani, L., Falcidieno, B., Nagy, G., and Pienovi, C., 1984. A hierarchical structure for surface approximation, *Computers and Graphics*, 8(2), 183 - 193.

DeFloriani, L., 1989. A pyramidal data structure for triangle-based surface description, *IEEE Computer Graphics & Applications*, 9(2), 67-78.

Dennehy, T.G., and Ganapathy, S., 1982. A new general triangulation method for planar contours, *Proceedings of SIGGRAPH '82*, 69-74.

Dwyer, R.A., 1987. Faster divide-and-conquer algorithm for constructing Delaunay triangulations, *Algorithmica*, 2(2), 137-151.

Fekete, G., 1990. Rendering and managing spherical data with sphere quadtrees, *Proceedings of Visualizaion '90*, 176-186.

Fournier, A., and Montuno, D., 1984. Triangulating simple polygons and equivalent problems, *ACM Transactions on Graphics*, 3(2), 153 - 174.

Fowler, R.J. and Little, J.J., 1979. Automatic extraction of irregular network digital terrain models, *Proceedings of SIGGRAPH '79*, 199-207.

Garey, M. R., Johnson, D. S., Preparata, F. P. and Tarjan, R. E., 1978. Triangulating a simple polygon, *Information Processing Letters*, 7, 175-180.

Goodchild, M.F., 1989. Optimal tiling for large cartographic databases, *Proceedings of AUTO-CARTO 9* , 444-451.

Manacher, G.K. and Zobrist, A.L., 1979. Neither the greedy nor the Delaunay triangulation of a planar point set approximates the optimal triangulation, *Information Processing Letters*, 9, 31-34.

Mirante, A. and Weingarten, N., 1982. The radial sweep algorithm for constructing triangulated irregular networks, *IEEE Comuters Graphics & Applications*, 2, 11-21.

Preparata, F.P. and Shamos, M.I., 1985. *Computational Geometry*, Springer-Verlag, New York.

Scarlatos, L.L., 1989. A compact terrain model based on critical topographic features, *Proceedings of Auto-Carto 9*, 146-155.

Scarlatos, L.L., 1990(a). An automated critical line detector for digital elevation matrices, *Proceedings of the 1990 ASPRS/ACSM Annual Convention*, 43-52.

Scarlatos, L.L., 1990(b). A refined triangulation hierarchy for multiple levels of terrain detail, *Proceedings of the IMAGE V Conference*, 115-122.

Watson, D.F., 1981. Computing the n-dimensional Delaunay tessellation with applications to Voronoi polytopes, *The Computer Journal*, 167-172.

Watson, D.F. and Philip, G.M., 1984. Survey: systematic triangulations, *Computer Vision, Graphics, and Image Processing*, 26, 217-223.

STRUCTURING THE KNOWLEDGE OF CARTOGRAPHIC SYMBOLIZATION - AN OBJECT-ORIENTED APPROACH

Feibing Zhan
Center for Computer Graphics and Mapping
Faculty of Geodesy, Delft University of Technology
Thijsseweg 11, 2629 JA Delft, The Netherlands

ABSTRACT

Knowledge-based systems for cartographic symbolization concerned with GIS's output have been suggested by a number of researchers. The structuring of the knowledge with a proper knowledge representation scheme is one of the key issues for the development of such a system. The specific requirements for the knowledge representation scheme are specified. It is argued that the conventional knowledge representation schemes such as rules, semantic networks, conceptual graph, object-attribute-value and frames are not rich and powerful enough to meet the requirements. Then it is suggested to use an object-oriented knowledge representation (OOKR) scheme to construct the knowledge. It meets the requirements and overcomes major problems of the conventional knowledge representation schemes. Further, examples are given to demonstrate the power and flexibility of the object-oriented knowledge representation scheme.

1. Introduction

The analysis results of a GIS are usually represented by maps which are, at present, generated through the relevant facilities of a GIS automatically or interactively. The maps are subsequently used as a major tool for decision making and communication. Currently, none of the GIS systems includes mechanisms to ensure the correct use of graphic functions. This may lead to poor use of graphics as GIS systems are widespread, and many of the users of GIS's are not professional cartographers. Indeed, many poorly designed maps can be observed (Muller and Wang, 1990). To solve this problem, considerable investigations on using knowledge-based system technology have been conducted and some achievements have been made (Mark and Buttenfield, 1988; Muller and Wang, 1990; Weibel and Buttenfield, 1988). However, no comprehensive and truly intelligent system has been constructed up to now.

Many issues should be addressed for developing a full-scale map design knowledge-based system (Mackaness and Fisher, 1986; Weibel and Buttenfield, 1988). Among these issues, a proper knowledge representation scheme that can be used to organize the relevant knowledge and facilitate the relevant issues concerned is fundamental for the development of such a system. In the Artificial Intelligence (AI) community, commonly used knowledge representation schemes are rules, semantic networks, conceptual graph, object-attribute-value(OAV) and frames. Each of them has certain advantages and disadvantages. Muller and Wang (1990) used a frame-based knowledge representation scheme for cartographic symbol design. Wang (1990) proposed a conceptual graph based representation scheme for cartographic information representation.

In this paper, it is suggested to use object-oriented knowledge representation (OOKR) scheme for a knowledge-based system for cartographic symbolization concerned with GIS's output (hereafter we will only call it cartographic symbolization). In next section, the specific requirements for the knowledge representation scheme are specified. In Section 3, it is argued why the conventional schemes are not rich and powerful enough to meet the requirements, and why object-oriented knowledge representation scheme is suitable. The representation of the knowledge of cartographic symbolization by the object-oriented representation scheme is illustrated by examples in Section 4. Discussions and future work are given in Section 5.

2. The Requirements for the Knowledge Representation Scheme

A knowledge representation scheme is the way in which the facts and relationships of the domain knowledge are organized. It is an issue of key importance for developing a knowledge-based system. General requirements of a knowledge representation scheme can be found, for example, in Luger and Stubblefield (1989). Up to now, there has been no comprehensive knowledge representation scheme which can be used to organize every kind of knowledge. The choice of the knowledge representation scheme depends on the characteristics of the domain knowledge under consideration. The first question then is: what are the requirements of the knowledge representation scheme for cartographic symbolization?

First, let us have a look at an example. Suppose a geographic information system contains information about the buildings of a municipality. A user of the system wants to have the statistical information of each district about area of buildings used for residential and industrial purposes respectively, and the statistic information must be represented on a map. The common procedure for the generation of the map, at present, is: First, one groups the two kinds of information on each district (e.g. by SQL) to produce a data file. Second, one designs the map type and relevant symbols for representing the information based on cartographic symbolization principles. In this step, besides the rules used for decision making, some calculation is often necessary, for instance, to determine the value and size of a symbol. Then the designed map parameters are passed to a package (e.g. GIMMS) to generate the map. If a cartographic symbolization knowledge-based system is attached to the GIS, it is natural and desirable that the knowledge representation scheme could facilitate the issues concerned with the three steps mentioned above.

More generally, the following issues are essential requirements for a knowledge representation scheme when developing a knowledge-based system for cartographic symbolization.

a. Like any knowledge representation scheme, the scheme must have the capabilities to describe the objects and model the relationships between the objects concerned with cartographic symbolization. The objects in cartographic symbolization are those concerning the interpretation of the spatial information to be mapped, the cartographic symbolization principles, and the relevant cartographic semiology.

b. An important feature of spatial information, and the relevant cartographic symbolization principles is their organization into class hierarchies (e.g. Egenhofer and Frank, 1990; Muller and Wang, 1990). Thus the ability of the knowledge representation scheme to represent the inheritance between a class and its instance objects, and between a class and its superclass is essential.

c. As it is believed that the development of a map design knowledge-based system should be started from a limited domain (Muller and Wang, 1990), and thus knowledge may then be gradually acquired in an "amplified intelligence" strategy (Weibel and Buttenfield, 1988), it is desirable that the knowledge representation scheme should be well structured and be able to support modularity and reusability. Hence, when the size of the knowledge base increase significantly, the knowledge is still manageable, and can be extended and reused.

d. When the knowledge base grows and changes, consistency checking becomes important. Moreover, judging from the issues concerned with map design knowledge-based systems (e.g. Muller and Wang, 1990), one can see that map design and generation are problems that mix logical deduction, rule-based inference, and procedure execution (e.g. graphics generation). The solution of these problems demands a knowledge representation scheme that effectively combines rules and procedures, and provides a vehicle for implementing graphics I/O, consistency checking, and interactions between objects.

e. When using GIS, spatial information to be mapped is usually from the database of a GIS, this information is then used for deduction, reasoning and map generation. Therefore the knowledge representation scheme should not only be able to support data input through consultation, but also be able to facilitate automatic feeding of data from a spatial database. This should be considered as an essential feature of the knowledge representation scheme.

These issues may be partially addressed by combining existing technologies such as database, conventional knowledge representation scheme and mapping packages (e.g. Muller and Wang, 1990). However, what is desirable is that the issues could be accommodated by a knowledge representation scheme in a uniform way. We will see how object-oriented approaches can be used to facilitate the issues.

3. Why an OOKR Scheme is Suitable for Structuring the Knowledge

We will see, in this section, why the conventional knowledge representation schemes can not meet the requirements discussed in Section 2, and describe the promises of the object-oriented knowledge representation scheme.

3.1 Object-oriented knowledge representation

Follow Luger and Stubblefield (1989), and Meyer (1988), an object-oriented knowledge representation scheme may be defined as the organization of knowledge as structured collections of abstract data type implementations. In this scheme, everything is defined as an object or system of objects. An object can be defined as an independent entity represented by some declarative data and a set of methods (such as routines and rules) that operate on the object. Relationships between objects and the overall problem specification are implemented as messages between objects. In addition, objects are abstracted into a hierarchy of classes, allowing the inheritance of properties and methods.

For other basic concepts concerned with object-oriented knowledge representation such as classes, inheritance, attributes, methods, controls, message passing, encapsulation, redefinition, polymorphism, dynamic binding, modularity and reusability, we refer to Leung and Wong (1990) and Meyer (1988).

It should be noted that object-oriented knowledge representation scheme is different from conventional knowledge representation schemes (except frames) in that knowledge is abstracted to classes which are instantiated by objects. It differs from commonly-called object-oriented approach for software construction in that rules are included in methods.

To adequately model a complex system in reality, abstraction mechanisms are necessary. The fundamental abstraction mechanisms from the database paradigm can be used. These abstract mechanisms are classification, generalization and aggregation (Smith and Smith, 1977). Classification is the abstraction from individuals with common properties and behavior to a class, by which 'instance-of' relation is modeled. Generalization is the combination of several classes to a more general superclass, by which 'is-a' relation is modeled. A class that references one or more other classes is called an aggregation of those other classes. By using aggregation, a 'has-a' relation between classes is modeled. Using types in the various relations and message passing, any kind of specific relations can be modeled (Meyer, 1988).

3.2 Conventional versus object-oriented knowledge representation

Conventional knowledge representation schemes, such as rules, semantic networks, conceptual graph, object-attribute-value triples and frames, are commonly used in traditional knowledge-based systems (Luger and Stubblefield, 1989; Townsend, 1986). Each of them has its own advantages and disadvantages (Leung and Wong, 1990).

As pointed out by Leung and Wong (1990), a common shortcoming in rules, semantic networks, conceptual graph and OAV representations is that they are not structured enough. Because the knowledge cannot be modularized, the interactions among rules and objects become too complex when the number of objects or rules in the system increases significantly. Thus the system becomes very difficult to manage. When the value of an attribute is modified, it is difficult to pinpoint the effects on the whole system. Therefore, such knowledge representations are difficult to develop and maintain, especially for a large knowledge base like cartographic symbolization.

Frames are more structured than rules, semantic networks, conceptual graph and OAV knowledge representations, since related attributes and rules can be grouped into frames hierarchically. However, modularity of knowledge represented in frames can not be clearly defined, and frame representation lacks flexibility. In a frame system, relationships between frames may be member or subclass links and thus are not unique. Moreover, in some systems, a rule is represented by a frame linked to another frame with special relationship. These factors greatly reduce the structure in a frame system (Leung and Wong, 1990).

Another shortcoming of the conventional knowledge representation schemes is that the objects represented in the schemes are not active. Thus operations through message passing between objects are not possible. Although frames allow the creation of complex objects and the integration of procedural and declarative representations, they are passive data structures that must be acted on by external procedures. The execution of attached procedures requires that the procedure definition be retrieved and evaluated by some external agent (Luger and Stubblefield, 1989).

Object-oriented knowledge representation scheme has the following advantages over the conventional schemes.

Firstly, like semantic networks and conceptual graph, it is flexible. In object-oriented knowledge representation, by storing the names of other objects as the attributes of an instance object, relations between instance objects can be established dynamically (Leung and Wong, 1990). These relationships have the same power as links in semantic networks, and relationships in conceptual graph. In fact, the object-oriented construct can be viewed as dynamic semantic network. The 'is-a' links of semantic network can be implemented in object-oriented representations by relationships between classes and subclasses or between classes and instances. The 'has-a' links can be implemented by the relationships between classes and attributes.

Secondly, object-oriented knowledge representation supports classes and inheritance. In a pure object-oriented system, everything is an object; all objects are abstracted to a certain number of classes. This allows inheritance of attribute names, values, and methods. In addition, each class defines instance variables, which must be instantiated when an individual member of that class is created. Instance objects bind these variables to all the particular information, such as size and location, that distinguishes individuals from each other. The behavior of the members of the class, or the set of all messages to which the class responds, is called the protocol of the class (Luger and Stubblefield, 1989).

Thirdly, it supports modularity and reusability. Modularity and reusability are of prime importance for any truly flexible system. A true modularized system should facilitate modular decomposability, modular composability, modular understandability, modular continuity and modular protection. To achieve these modular capabilities, modules must correspond to syntactic units in the language used, every module should communicate with as few others as possible, exchange of information between modules should be as little as possible, interfaces between modules must be explicit and all information about a module should be private to the module unless it is specifically declared public. Five issues must be solved before we can hope to produce practically reusable modules. These issues are: variation in types, variation in data structure and algorithms, related routines, representation independence and commonality within subgroups (Meyer, 1988). Object-oriented approach satisfies the criteria and principles of modularity, and provides a remarkable set of answers to the set of reusability issues (Meyer, 1988).

Finally, declarative and procedural knowledge can be integrated, and the objects are active. Objects in a object-oriented knowledge representation scheme are active in the sense that the methods are bound to the object itself, rather than existing as separate procedures for the manipulation of a data structure. Objects thus have characteristics of both data and programs in that they retain state variables as well as react procedurally in response to appropriate messages. Objects execute their methods directly in response to a received message. It is the active nature of objects that makes the message passing, execution of methods (rules, routines, etc.) possible. Such methods provide the vehicle for consistency checking, implementing graphics I/O, and combining rules and procedures.

3.3 How OOKR scheme facilitates the requirements

Based on the observations in the last two subsection, we then discuss how the OOKR scheme facilitates the specific requirements which are specified in Section 2.

a. Objects and their relationships can be represented in both passive form, and active form by a mixture of attributes, rules, routines, 'is-a' relations, 'has-a' relations and messages. Therefore declarative and procedural knowledge can be integrated in a uniform way, and complex knowledge can be adequately organized.

b. Inheritance exists between classes and subclasses. Thus, knowledge can be represented in an abstracted form with common features generalized in a superclass. Existing classes can be extended and reused by using relevant techniques in object- oriented approaches.

c. As object-oriented approach facilitates modularity, related rules can be well grouped in a class or a module that is independent of other classes or modules. This enhances manageability, understandability and maintainability.

d. Rules and procedure executions can be defined in methods, thus rules and procedures are naturally combined. Routines can be defined by any language which produces routines in an executive form, and then bound to the objects, hence routines such as graphics generation and parameter calculation can be conveniently performed.

e. Data can not only be input through consultation but also be automatically feed from a spatial database through the execution of relevant methods, therefore a knowledge-based system based on this scheme can be naturally attached to a GIS.

Hence it can be concluded that the object-oriented knowledge representation scheme provides a set of answers to the specific requirements, and gives the promises to fully address the issues concerned with cartographic symbolization in a uniform way.

4. Examples of Knowledge Structuring for Cartographic Symbolization

In this section, first an example is used to demonstrate how object-oriented knowledge representation scheme can be used to address the issues concerned with cartographic symbolization. Then the abstraction of the knowledge, the capability of the scheme to support reusability and extendibility are discussed.

4.1 Knowledge structuring of cartographic symbolization for representing statistical building information from GIS - the example

Let us see how object-oriented knowledge representation scheme can be used to address the issues concerned with the example mentioned in section 2. To solve the problem, classes 'building', 'statistical_map', 'graphics_map' are defined. The definition of each class is illustrated as below. For the convenience of illustration, the definition of the classes is condensed. The notation used is those from Luger and Stubblefield (1989), except that rules are also included in the methods.

```
Class name: building
Superclass: ....
Instance variables: district_identifier, building_identifier,
    building_type, area, ...
Instance methods: ...
    group(): begin
              message(district_identifier, building_type, total_area)
        end
  ...
  end
Class methods: ...


Class name: statistical_map
Superclass: thematic map
Instance variables: map_type, info_property, title, legend, ...
Instance methods:
     info_input(): begin
         for i:= 1 to N do begin
            message(info_property(i))
        end
     end
  end
    map_type():    begin
         rule:  IF info_property = <quantitative> & <absolute> & <multiple>
                THEN map_type = <graphics_map>
                end
        ...
        end
        ...
Class methods:
   begin
    message(map_type, symbol)
      map_generation(title, map_type, symbol, legend)
  end


Class name: graphics_map
Superclass: ...
Instance variables: symbol_type, no_of_variables, variable_color,
    variable_size, ...
Instance methods:
        symbol_type():  begin
                rule:  IF map_type = <graphics_map>
                        THEN symbol_type = <bar_graphs> or <pie_charts>
                        end
                ...
        end
        no_of_variables():  begin
                        message(no_of_variables)
                        end
        end
        variable_color():  begin
                for i:=1 to <no_of_variables> do begin
                message(variable_color(i), color)
                        end
                end
        end
        size(): begin
                for i:=1 to <no_of_variables> do begin
                message(variable_size(i),  size_calculation)
                        end
                end
        end
        ...
    Class methods: ...
```

After these classes have been defined, the cartographic symbolization can be effected by message passing. Through assembling the classes together by message passing, building information can be grouped from relevant database; consultation can be conducted; the map type can be inferred (graphic maps); the symbol can be determined (bar graphs) and the visual variables can be calculated (two elements, color and size), and finally the statistical map can be generated.

It is easy to see from the example that all issues such as procedures, rules, data (described by attributes) can be addressed in a uniform way with the scheme. But, this simple example can not completely show the power of the scheme. The power of the scheme is the abstraction of complex knowledge, the capability to support extendibility and reusability for constructing a system in the large, in our case, the construction of knowledge-based system for cartographic symbolization. We discuss these issues in the following sub-section.

4.2 Abstraction, extendibility and reusability

Although the above example has shown the flexibility and power of the object-oriented knowledge representation scheme, one can not see from it how the complexity of the knowledge of cartographic symbolization can be modeled. In this section, we will first discuss how the knowledge of cartographic symbolization can be abstracted by classification, generalization and aggregation. We will then illustrate the extendibility and reusability of the abstracted knowledge.

Aspects concerned with cartographic symbolization are generally the interpretation of spatial information concerned, the choice of map type and the design of symbols. These aspects can be sketched as in Figure 1.

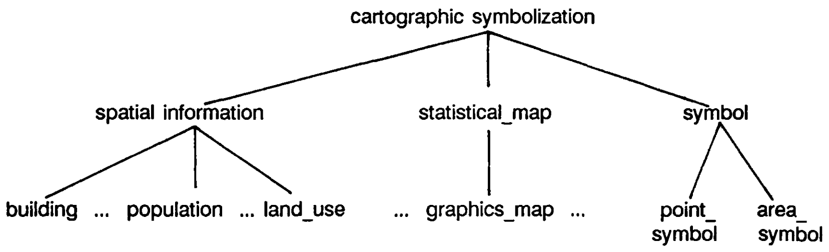


Figure 1 Sketched aspects of cartographic symbolization

To model these aspects in an abstract form, abstraction mechanisms are often used. One can see from Figure 1, information contained in a GIS may be classified as 'building', 'population', 'land use' and so on; these classes can then be generalized to a superclass- 'spatial information'. Likewise, map symbols are often classified as 'point symbol' and 'area symbol', the common property of these classes can be generalized in an abstract form in a superclass - 'symbol'.

Let us take the symbol module and go into depth. All point symbols can be considered as instance objects of class 'point_symbol' which is defined as follows:

```
Class name: point_symbol
Superclass: symbol
Instance variables: form, orientation, color, texture, value, size
Instance methods:
        form():  begin
                message(form, circle)
                ...
                end
        end
        orientation():  begin
                message(orientation, 45)
                ...
                end
        end
        color():  begin
                message(color, green)
                ...
                end
        end
        texture():  begin
                message(texture, 1/5)
                ...
                end
        end
        value():  begin
                message(value, value_calculation)
                ...
                end
        end
        size():  begin
                message(size, size_calculation)
                ...
                end
        end
Class methods: ...
```

Class 'point_symbol' can be regarded as subclass of class 'symbol'. By generalization, the above definition can be revised as follows:

```
Class name: symbol
Superclass: ...
Instance variables: orientation, color, texture, value
Instance methods:
        orientation(): begin
                message(orientation, 45)
                ...
                end
        end
        color(): begin
                message(color, green)
                ...
                end
        end
        texture(): begin
                message(texture, 1/5)
                ...
                end
        end
        value(): begin
                message(value, value_calculation)
                ...
                end
        end
Class methods: ...


Class name: point_symbol
Superclass: symbol
Instance variables: form, size
Instance methods:
        form(): begin
                message(form, circle)
                ...
                end
        end
        size(): begin
                message(size, size_calculation)
                ...
                end
        end
Class methods: ..
```

The above two definitions are abstractive in the sense that any point symbol can be generated through the definitions. They are generalized because common visual variables such as orientation, color, texture and value are defined in the superclass 'symbol'. This ensures that common behaviors across several subclasses (point and area symbols) will indeed have common definition, and instance variables and methods in class 'symbol' can be inherited by its subclass 'point_symbol'.

We then discuss how the definitions can be extended and reused. Suppose that a knowledge base only contain the above two classes about symbol, and now one wants to add class 'area_symbol' into the knowledge base. The question becomes how the definitions can be reused and extended without modifying the existing two classes. In this case, the answer is very simple:

use inheritance and redefinition to define a new class 'area_symbol' as illustrated below. In the class 'area_symbol', 'color', 'texture' and 'value' are redefined. Only orientation is inherited from the superclass 'symbol'.

```
Class name: area_symbol
Superclass: symbol
Instance variables: color, value, texture
Instance methods:
        color(): begin
                for i:=1 to N do begin
                message(color(i), color)
                end
                end
        end
        texture(): begin
                for i:=1 to N do begin
                message(texture(i), texture)
                end
                end
        end
        value(): begin
                for i:=1 to N begin
                message(value(i), value_calculation)
                end
                end
        end
Class methods: ...
```

After the examples, one can immediately see that knowledge concerned with the interpretation of spatial information and determination of map type can be abstracted in a similar way. And then can be reused and extended by using inheritance, redefinition, polymorphism and dynamic binding (Meyer, 1988).

5. Discussion and Further Work

Based on the specification of the requirements of the knowledge representation scheme for representing the knowledge of cartographic symbolization concerned with GIS's output, it is argued that the conventional knowledge representation schemes such as rules, semantic networks, conceptual graph, object-attribute-value and frames are not rich and powerful enough to meet the requirements. Then it is suggested to use the object-oriented knowledge representation scheme to represent the spatial knowledge concerned with cartographic symbolization. Discussions show that the object-oriented approach meets the specific requirements and overcomes the major problems of the conventional schemes.

The flexibility and the power of the OOKR scheme are only partially illustrated with examples. The work reported in this paper is still far from fully structuring the comprehensive knowledge of cartographic symbolization. However, as an approach, the object-oriented knowledge representation scheme offers greater potentials for capturing, organizing, processing the knowledge and applying it in the digital domain.

Once a reasonable amount of knowledge is specified and structured with the scheme, the knowledge base and inference engine can be implemented by a suitable media (e.g. a suitable knowledge-based system shell supporting object-oriented knowledge representation). Our opinion is that the object-oriented approach in general is a whole paradigm, in which object-oriented analysis, object-oriented design and object-oriented programming can be distinguished (see e.g. Coad and Yourdon, 1990). As far as only analysis and design (in this case high level knowledge structuring) are concerned, the object-oriented knowledge representation scheme is regarded as a high level construct.

To fully structure the knowledge of cartographic symbolization, a number of issues are still subject to further investigation.

Firstly, further investigation on the object-oriented knowledge representation scheme itself is still necessary, for example, multiple inheritance, and semantics to ensure correctness and robustness. These are the particular interests of the author and will be investigated in the near future.

Secondly, the interpretation of spatial information from a GIS should be addressed in detail as it is the fundamental step for the subsequent symbolization. Several issues are concerned with this aspect, for example, the encapsulation of knowledge in spatial database (this is effected through methods in the OOKR scheme), the rules for the interpretation of the spatial information, and the relationships between the two. These issues are currently under investigation.

Thirdly, much work needs to be done to use the scheme to structure the comprehensive knowledge of cartographic symbolization. To address this, the comprehensive knowledge concerned should be specified first. After sufficient knowledge is specified and captured, the knowledge then can be abstracted into classes by using the object-oriented knowledge representation scheme. The complicated relationships can be represented by 'is-a' relations, 'has-a' relations and messages. New knowledge can be captured gradually, and added to the knowledge base by defining new classes and/or subclasses of existing classes. A full scale knowledge-based system for cartographic symbolization then can be eventually achieved.

6. Acknowledgements

I sincerely thank Prof. Dr. Ir. M.J.M. Bogaerts for his stimulating discussions and guidance, and for his arrangement which made this investigation possible. Almost all of our colleagues in the Group of Land Information and Cartography, Faculty of Geodesy, Delft University of Technology, read an earlier version of this paper and gave their comments; I wish to thank all of them. Valuable comments from the anonymous referees are gratefully acknowledged.

7. References

Coad, P. and E. Yourdon, 1990, Object-Oriented Analysis, Yourdon Press Computing Series.

Egenhofer, M.J. and A.U. Frank, 1990, LOBSTER, Combining AI and Database Techniques for GIS. Photogrammetric Engineering and Remote Sensing, Vol. 56, No. 6, pp. 919-926.

Leung, K.S. and M.H. Wong, 1990, An Expert System Shell Using Structured Knowledge - An Object-Oriented Approach, IEEE Transactions on Computer, Vol. 23, No. 3, pp.38-47.

Luger, G.F. and W.A. Stubblefield, 1989, Artificial Intelligence and the Design of Expert Systems, Benjemmin/Cummings Publishing Company, Inc.

Mackaness, W.A. and P.F. Fisher, 1986, Towards a Cartographic Expert System, In Proceedings of Auto Carto London, pp.578-587.

Mark, D.M. and B.P. Buttenfield, 1988, Design Criteria for Cartographic Expert System, In Proceedings of the 8th International Workshop on Expert Systems, Avignon, France, Vol.2, pp.413-425.

Meyer, B., 1988, Object-Oriented Software Construction, Prentice-Hall.

Muller, J.-C. and Wang Zeshen, 1990, A Knowledge Based System for Cartographic Symbol Design, The Cartographic Journal, Vol. 27, No. 2, pp. 24-30.

Smith, J.M. and D.C.P. Smith, 1977, Database Abstractions: Aggregation and Generalization, ACM Transactions on Database Systems, Vol. 2, No.2, pp.105-133.

Townsend, C., 1986, Mastering Expert Systems with Turbo Prolog, Howard W. Same & Company.

Wang, Z.S., 1990, A Representation Scheme for Cartographic Information, In Proceedings of the 4th International Symposium on Spatial Data Handling, Zurich, pp. 782-791.

Weibel, R. and B.P. Buttenfield, 1988, Map Design for Geographic Information Systems, In Proceedings of GIS/LIS'88, San Antonio, Texas.

Are Displays Maps or Views?

WERNER KUHN
National Center for Geographic Information and Analysis
and Department of Surveying Engineering
University of Maine
Orono, Maine 04469 (USA)
Bitnet: Kuhn@mecan1

Abstract

Metaphors are powerful means to design and learn user interfaces for computer systems. This paper discusses metaphors for display operations in Geographic Information Systems (GIS). Specifically, the metaphor DISPLAYS ARE VIEWS is proposed and analyzed. It is presented as an antithesis to the metaphor DISPLAYS ARE MAPS, which is consciously or unconsciously adopted by designers and users of most GIS interfaces. Displays are understood here as graphic screen presentations of geographic space, maps as static (paper) maps and views as visual fields, containing what humans see in a given situation. The major advantage of the visual field as a metaphor source is that it naturally accommodates scale changes. Thus, analyzing its structure also sheds new light on the generalization problem for displays.

1. Introduction

Metaphors have had a significant impact on general user interface design practice and are now established as a powerful means to control complexity in human-computer interaction [Carroll, Mack, and Kellogg 1988]. Their potential for improving user interfaces of Geographic Information Systems (GIS) is also rapidly gaining recognition, as indicated by a series of recent publications dealing with the subject [Gould and McGranaghan 1990, Jackson 1990a, Kuhn 1990, Mark 1989, Wilson 1990]. A common theme of these studies is the selection of appropriate metaphors for GIS user interfaces. Currently, map metaphors dominate, but it has been suggested that they fail to organize GIS operations adequately [Gould and McGranaghan 1990].

This paper discusses the metaphor question for GIS display functions, where the map idea is least controversial and most entrenched, as exemplified by the common expressions "virtual map", "screen map", or "CRT map". The paper contends that map metaphors are deficient even for display purposes and proposes the contrasting metaphor DISPLAYS ARE VIEWS. It shows that human vision provides a rich and powerful source of metaphors for retrieving and displaying information. In particular, it focuses on the capacity of the visual system to deal with resolution and scale changes.

Research in cognitive science has established that humans perceive, conceptualize and deal with the world at multiple levels of detail [Marr 1982, Minsky 1985]. A GIS should support this capacity, by representing data at multiple resolutions and offering operations appropriate to scale [Buttenfield and Delotto 1989]. Yet, while there has been considerable interest in database representations and manipulations at multiple levels of resolution [Guptill 1989, Oosterom 1991, Samet 1989], the same cannot be said for user interface representations.

Cartographers and GIS specialists are still struggling for a satisfactory understanding of the concepts of scale and resolution. There appear to be two dominating lines of thought: the "pragmatists" understand resolution in terms of map scale, acknowledging the limits of this concept, and the "objectivists" look for geographic scale or dimensions in the real world.

Since resolution is also a concept of human vision [Marr 1982], a third way could be to explain scale in terms of vision and its properties. Such an "experientialist" approach [Lakoff 1987] based on human perception of and interaction with the world [Arnheim 1969] is taken here. Specifically, the fundamental relation of scale and scale changes to viewing distance is explored. The goal is to apply this elementary human experience to GIS user interfaces through metaphors.

The remainder of the paper contains a discussion of interface metaphors for GIS in section two, preparing for an analysis of the DISPLAYS ARE VIEWS metaphor in section three, after which conclusions are drawn and further work is suggested in section four.

2. Metaphors and GIS interfaces

2.1. Metaphors and image-schemas in human-computer interaction

Johnson [1987, p. XIV] has characterized metaphor as

> *...a pervasive mode of understanding by which we project patterns from one domain of experience in order to structure another domain of a different kind.*

The two domains are commonly called the *source* and *target* domains of a metaphor and the metaphorical projection can be seen as a *mapping* (in the mathematical sense) from source to target. Johnson's characterization expresses a projective view of metaphor: the metaphor *imposes* a structure on the target domain, rather than assuming similarities between source and target.

Lakoff and Johnson have argued convincingly that ordinary (i.e., non-poetic) thought, action, and language are structured by metaphor [Lakoff and Johnson 1980]. It seems reasonable to presume that this is true for thought, action, and language in human-computer interaction as well. Interface metaphors are doing far more than just helping novices to learn a new application. They structure the application domain and organize the user's tasks. The designer's choice of metaphor(s) determines what concepts the users will have to deal with, how the labor is distributed between users and system, and in what terms users and system will communicate.

Since metaphorical projections can be described as mathematical mappings between domains, what remains invariant under them? Lakoff's invariance hypothesis [Lakoff 1990] claims that it is the image-based reasoning patterns of the source domain, the so-called image-schemas [Johnson 1987, Lakoff and Johnson 1980]. These are idealized cognitive structures, consisting of a small number of parts and relations, made meaningful by human sensori-motor experience. Examples are the CONTAINER, PATH, LINK, NEAR-FAR, PART-WHOLE, and CENTER-PERIPHERY schemas. Image-schemas are more abstract than mental images, being essentially reduced to topology, but less abstract than logical propositions, being related to sensori-motor experience.

It has been suggested that image-schemas play a fundamental role in user interfaces and that they are likely to be especially relevant for GIS interfaces, since many image-schemas are spatial, particularly topological, in nature [Mark 1989]. General GIS metaphors are further discussed by Gould and McGranaghan [1990]. An extended discussion of the role of metaphors and image-schemas in user interfaces, including a formalization, can be found in [Kuhn 1991].

2.2. Map metaphors and GIS

Most of today's GIS interfaces have been designed explicitly or implicitly with (hardcopy) maps and mapping operations in mind. Consequently, mapping concepts dominate the whole spectrum of GIS functions, from data acquisition through analysis to display.

Some generic problems with map metaphors have been discussed in the literature [Downs 1981, Gould and McGranaghan 1990]: Maps may not be understood well enough to serve as a useful source domain, they provide little guidance beyond display operations, they tend to hide uncertainty in the data, and they are two-dimensional representations of a three- or four-dimensional reality.

At any rate, maps are unlikely to be adequate sources of GIS metaphors for *all* the different kinds of functions which paper maps fulfill, serving at the same time as data storage and presentation devices, and as analysis and design tools. For example, maps and map sheets are now widely recognized as inappropriate analogues for the data storage function of a GIS. The main reason is that maps lead to undesirable partitionings of data, both horizontally (sheets) and vertically (layers) [Chrisman 1990, Frank 1988]. The evolution from layered mapping systems to seamless geographic databases with integrated topological data structures is practical evidence for this movement away from the map metaphor in data storage.

What about data presentation functions? GIS displays are generally understood as "screen maps", implying the metaphors DISPLAYS ARE MAPS and DISPLAYING IS

MAPPING. Thereby, they inherit not only useful conventions on symbolisms and the goal of graphic excellence, but also some limitations and problems. For instance, paper maps handle multiple resolutions in a rigid way through series of scales and pose the difficult problem of cartographic generalization, i.e. adapting information and its presentation to scale [Brassel and Weibel 1988]. While many aspects of this problem will also have to be dealt with for displays no matter what metaphors are chosen, it is worth looking for possible differences between requirements for maps and displays.

One way to do this is by asking how the visual system copes with generalization: For example, why does one never see a cluttered world (at least not in the sense of a cluttered map or display)? Controlling data density, one of the hardest problems in generalization, seems no problem at all in human vision. Understanding how the visual system achieves this could help solving the problem for displays. Also, objects which are too small to recognize are acceptable in visual fields: we ordinarily see things of which we cannot make sense because they are too small. On maps, such unrecognizable objects are not tolerable. Displays as well as views, however, can allow users to "zoom" in and see more detail (see 2.4. for more detail on this).

2.3. Visual interfaces and GIS

Clearly, electronic screens offer far more possibilities for GIS data presentation than paper maps [Moellering 1984, Robertson 1988], despite their yet inferior resolution. For example, they allow for reactive, dynamic, and three-dimensional displays [Goodchild 1990]. Thereby, an entirely new kind of communication about geographic phenomena becomes possible, where users can interact directly with suitable and adaptive representations of these phenomena [Mark 1989].

This direct communication between user and system is not limited to the visual channel; non-visual means are rapidly gaining importance [Negroponte 1989]. State-of-the-art user interface technology, however, favors visual over auditory and tactile interaction. GIS interfaces are generally not disadvantaged by this emphasis, given the highly spatial nature of vision.

It is well established by now that seeing is more than passive perception [Arnheim 1969] and typically involves categorizing what is seen [Lakoff 1987]. An entire chapter of "The Nature of Maps" [Robinson and Petchenik 1976] is devoted to the discussion of how theories of visual perception and cognition relate to geographic data presentation. It emphasizes that our visual system is not a neutral input device and that seeing is an active process: we make sense of what we see by attempting to construct meaningful shapes.

The notion of "visual interfaces" [Tauber 1987] implies such an active involvement of the user. Apart from pointing gestures and actions like "dragging" [Apple Computer 1987], visual interfaces often contain metaphors related to special visual experiences like seeing through frames, lenses, and other optical instruments. Examples of these metaphors are "windows" [Smith et al. 1982], "panning" and "zooming" [Jackson 1990a], or "fisheye views" [Furnas 1986].

The widespread occurrence of these viewing metaphors suggests a more literal interpretation of the notion of "visual interfaces", exploring metaphors based on the human visual system as such, independent of optical instruments. The visual field not only offers the logic and functionality expected from displays - being a bounded, connected region which can be moved to see something else - it also deals very effectively with changes of scale (see 3.1.).

Geometric aspects of visual perception have been discussed, for example, by [Marr 1982, Zeeman 1962] or, in relation to geography, by [Tobler 1976]. For metaphors based on vision, the effects of these geometric properties on visual cognition are of interest. An important case of such an effect is the phenomenon that, by moving closer to a scene, we not only get to see enlarged objects, but *different kinds* of objects. For example, we may see a house across the street as consisting of walls, windows, a door, and a roof and from its front yard, we can identify individual planks and bricks in the walls, but don't see the house as a whole anymore (Figure 1).

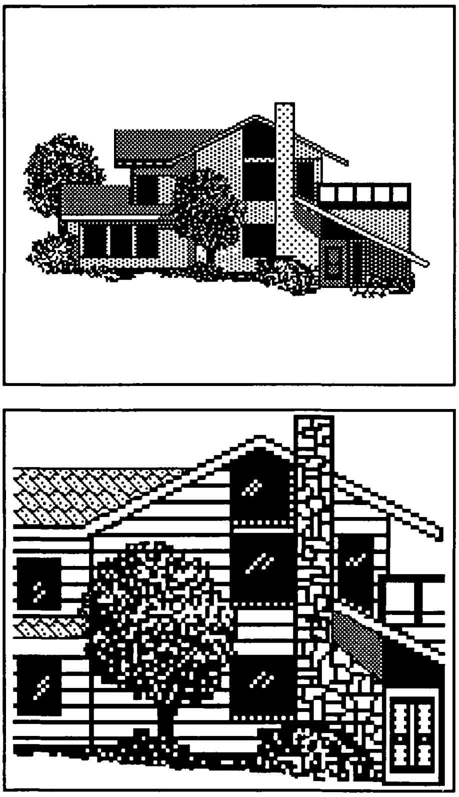


Figure 1: Getting to see different things by moving closer

This basic property of visual cognition, imposing lower and upper bounds on the level of detail perceived at a given viewing distance, is the source of many metaphors. In everyday language, it is often combined with the metaphor UNDERSTANDING IS SEEING to produce expressions like "let's take a closer look at this idea" or "he can't see the forest for the trees". In technical as well as colloquial language it is sometimes referred to as the "zoom" effect. The connection between levels of detail and the concepts of "close" and "distant" is also touched on in the final paragraphs of [Robinson and Petchenik 1976]:

"Scale" also refers to the level or depth with which one contemplates or analyzes something, as for example whether one "looks closely" at something or contemplates it "from a distance."

2.4. Zooming in on "zooming"

The term "zooming" is used in cinematography, photography, computer graphics and everyday language to describe getting "close-up" views of something. In the context of GIS, Burrough [1986, p. 79] states, for example, that "most graphics systems allow the user to zoom in and display an enlarged part of the database". This description leaves it open whether "an enlarged part" means "the same, but enlarged" or "a part of the database that becomes only visible at a larger scale", or both.

The notion of a "cartographic zoom" proposed in [Bjørke and Aasgaard 1990] applies the concept of zooming to the generalization of map displays. It implies that zooming allows a user to see different things at different scales, but the idea of zooming is not further explored.

Zooming and panning operations on digital images and map displays have been studied and described, independently from the generalization problem, by Jackson [1990b]. The main conclusion from this work was that intuitive and effective interface tools require a deeper understanding of zooming and panning than one in terms of cameras or other optical instruments.

The Oxford English Dictionary (second edition, 1989) defines the original meaning of "zoom" as follows:

To make a continuous low-pitched humming or buzzing sound;
to travel or move (as if) with a "zooming" sound; to move at speed, to hurry.

The use of the term in photography and cinematography is, thus, already doubly metaphorical: It explains the variation of the focal length by a (fictive) motion of "rapidly closing in on a subject" which, in turn, is metaphorically related to the corresponding sound effect. (Note that one of the key metaphors in visual interfaces is, therefore, rooted in auditory perception).

Combined with our visual experience that the viewing distance influences *what* we see, zooming naturally acquires a stronger interpretation than "seeing the same, but enlarged". It becomes a mechanism to change the scale or level of detail at which one perceives and conceptualizes the world or a computer model.

This understanding of zooming suggests the more general metaphor DISPLAYS ARE VIEWS, which also accommodates additional transformations of the visual field. One of them is "panning", i.e. moving the view to another part of a "panorama" without changing the level of detail. Since transformations of the visual field correspond to

> *... basic cognitive processes such as focusing, scanning, superimposition, figure-ground shifting, vantage-point shifting* [Lakoff 1988, p. 121]

they are ideal candidates for metaphor sources.

3. The metaphor DISPLAYS ARE VIEWS

3.1. Image-schematic structure

While a general notion of interface "views" has been around for some time [Goldberg and Robson 1981], the richness of visual fields as a source domain for interface metaphors has not yet been analyzed. The discussion of GIS "user views" in [Mark 1989] relates views to image-schemas, but concentrates on the notion of database views rather than displays.

In order to make the DISPLAYS ARE VIEWS metaphor applicable to user interface design, its image-schematic structure needs to be analyzed [Kuhn 1991]. Determining the image-schemas underlying views allows designers to define the functionality of display operations based on the metaphor.

The basic image-schema involved in the visual field is the CONTAINER schema [Lakoff 1987]. It structures the visual field as a bounded space, consisting of a boundary, an interior, and an exterior: In a first approximation, things are either in or out of sight and they come into or go out of sight.

The visual field has also a center of attention and a surrounding region. Thus, the CONTAINER schema is combined with the CENTER-PERIPHERY schema [Johnson 1987], which provides for distinguishing foveal and peripheral vision.

In addition, and less obviously, the visual field is structured by an interaction of the PART-WHOLE with the NEAR-FAR schema. We experience objects in the world as configurations of parts, forming wholes. Our perception has evolved so that it can distinguish these elements. Visual perception, in particular, requires motion of the body or of the objects to extend this distinction beyond the limited range of configurations present in one view: Getting certain parts into view involves moving nearer and vice versa. This connection is the essence of scale changes and of the zooming mechanism described above.

The combination of the PART-WHOLE and NEAR-FAR schemas enters the visual field in a second way: Moving nearer entails that only a part of the things one saw before remains within the visual field. Thus, the visual field shrinks with respect to the scene viewed. This property can be applied to simulate the relative motion of observer and objects in interactive zooming operations. By shrinking a frame of reference corresponding to the visual field, the user simulates a close-in motion (figure 2) after which the system displays a part of the scene at a larger scale.

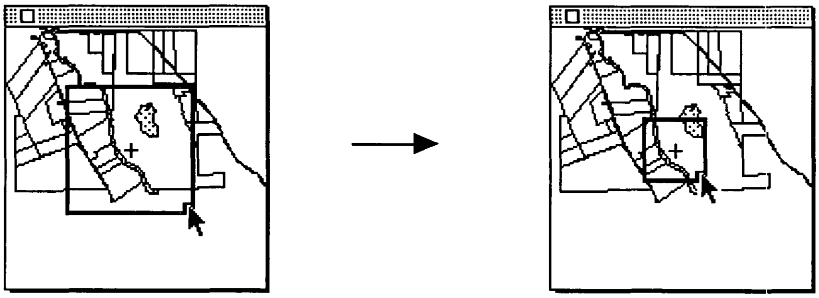


Figure 2: Combining the PART-WHOLE and NEAR-FAR schemas: Shrinking the visual field simulates motion in zooming operations. From [Jackson 1990b].

An interesting deviation from the "normal" behavior of containers defined by Lakoff and Johnson is that visual fields are not transitive: It is *not* necessarily true that, if A is in B and B is in sight, then A is in sight, too (even if one excludes the trivial cases where "in" means "enclosed by"). For example, if a wall is in sight and it is made from bricks, then it does not follow that individual bricks can be seen. Other examples are raster dots in an image or leaves in trees.

The superimposition of the PART-WHOLE and NEAR-FAR schemas on the CONTAINER schema explains this paradox: It allows a more specific interpretation of "A is in B" as "A is part of B" and lets the combination of PART-WHOLE and NEAR-FAR determine whether A is in sight or not.

DISPLAYS ARE VIEWS is a metaphor and not a literal equivalence. The matching between the two domains of a metaphor is by definition partial. Lakoff's invariance hypothesis suggests that some of the image-schematic structure in the two domains must correspond to support the metaphorical mapping. In the case of displays and views, this correspondence can be established for a combination of, at least, the CONTAINER, CENTER-PERIPHERY, PART-WHOLE, and NEAR-FAR schemas. Aspects which are non-topological and not image-schematic, such as the shape of the container boundary (elliptic vs. rectangular) or the type of optical projection involved (central vs. parallel), need not be equal or even comparable. Furthermore, some important features of displays, like symbolizations and their explanation, are obviously not accounted for by this partial correspondence with views.

3.2. Metaphor combinations and extensions

The DISPLAYS ARE VIEWS metaphor is both extensible and open to combinations with other metaphors in a GIS user interface. For example, GIS displays have to show properties and relations of phenomena which are invisible, such as borders, land values, or population densities. This situation requires the same additional metaphors as it does to explain why maps can represent them.

An extension of the visual field metaphor for displays is the previously mentioned idea of "fisheye views". Fisheye views of spatial phenomena are actually just an exaggeration of human views, which already have the property that the visual acuity lapses toward the periphery [Zeeman 1962]. This property provides a straightforward extension of the DISPLAYS ARE VIEWS metaphor. It has rarely been adopted for GIS displays, presumably because it violates the idea of a (roughly) constant scale inherent in the DISPLAYS ARE MAPS metaphor.

Fisheye views of non-spatial phenomena [Furnas 1986] and proposals for "conceptual" (logical, semantic) zooming [Mohan and Kashyap 1988, Tanaka and Ichikawa 1988] are all based on the same metaphor extension and combination: The idea that concepts are resolution dependent gets extended beyond spatial phenomena. The additional metaphor involved is that ABSTRACT SPACE IS PHYSICAL SPACE. Such an abstract space can, for example, be a hierarchy or a lattice. Thus it becomes possible to zoom in on an organization chart of a company from top-level divisions to individual workers.

Another direction of metaphor extension and combination leads beyond displays towards interface metaphors for *manipulating* GIS models. Such an approach is David Zubin's proposal to differentiate object classes based on object sizes relative to human experience [Mark et al. 1989]. Zubin discusses how shifting our viewpoint results in objects of different sizes becoming accessible to vision and manipulation. For example, a city which cannot be perceived as a unit when we are in it becomes a scene which we can scan when we drive away from it, and part of a single perspective when we are far away or flying over it. Zubin's classes of objects or spaces, thus, imply the notion of viewing distance and its influence on what object classes humans deal with. However, they are defined in terms of physical object sizes which become irrelevant for interaction with computer models. Further developing Zubin's ideas, Eric Bier has suggested editing paradigms for manipulating objects based on their relative sizes [Kuhn and Egenhofer 1991].

A promising extension of a perception-oriented understanding of displays are virtual realities [Conn et al. 1989]. GIS are probably the computing systems which come closest to dealing with actual three- or four-dimensional reality (although still too often only through two-dimensional projections). Thus, they should be ideal forerunners to systems which transcend the limitations of physical reality and allow users to experience motion through different scales by sight, sounds, and tactile cues, e.g. flying over territories or diving into atoms [Brooks 1988].

4. Conclusions

This paper has proposed human vision as a source domain for GIS interface metaphors. Specifically, it has argued for interfaces based on the metaphor DISPLAYS ARE VIEWS. The analysis of the image-schematic structure of visual fields, particularly of the fundamental connection between viewing distance and visible object classes, suggests that seeing is in some respects more powerful than mapping as a source domain for interface metaphors.

What does it mean to adopt the metaphor DISPLAYS ARE VIEWS? First, it involves the user in an active process of viewing rather than observing static maps. Second, it acknowledges the key role of the user's point of view in defining display contents. Here, "point of view" is still meant spatially. Social, political, and other viewpoints may, however, come to play an explicit role in future GIS applications. They are likely to fit this framework by direct metaphorical extension. Third, the metaphor allows the viewpoint to move conceptually closer to or further apart from a scene, supporting a notion of zooming which goes beyond magnification by relating different concepts to different scales.

WYSIWYG interfaces, where What You See Is What You Get, will clearly become more intuitive and more powerful when you have the kind of control over what you see that you have in ordinary visual experience. Spatial query and manipulation languages can become entirely different from today's awkward formalisms when they employ visual metaphors. A generalized zooming mechanism, for example, naturally integrates data retrieval and display operations. These are two aspects of querying which have been separated so far, in accordance with the idea that displays are maps, but to the disadvantage of the user [Egenhofer 1990].

For the sake of the argument, the notions of maps and views have been contrasted rather than integrated in this paper. An alternative approach to improved user interface metaphors would be to expand the notion of maps with visual concepts, taking today's display technology into account. It is equally valid and potentially leads to the same improvements. However, starting with visual concepts forces designers to evaluate more radically the role traditional mapping concepts should play in displays.

The contentions of the paper are not meant to imply that cartography has no role to play in visual displays. There are indeed many common concerns in the design of displays and maps and there is a lot to be learned from mapping and graphic excellence for data presentation on screens [Tufte 1983, 1990]. Arguing against the paper map as a dominating metaphor source is also not necessarily arguing against symbolization or against features like labels, legends, north arrows, and scale indications in displays.

The point made here is that there are important differences between requirements for good dynamic (display) and static (map) presentations. They are mainly due to the reactive character of electronic display media which supports a direct visual communication between user and system. These differences may allow for displays to relax some of the constraints on maps, like those on minimal dimensions and separations, which make automatic map generalization such a troublesome problem [Beard and Mackaness 1991].

It should be kept in mind that a metaphor such as DISPLAYS ARE VIEWS is never a literal equivalence. Thus, choosing human vision as a source domain does not restrict the scale range of displays to that of human views. The metaphor, establishing only a partial correspondence, takes some aspects of visual perception and uses them to structure displays. The possibility of zooming is one of these aspects, but restrictions of scale range, perspective, and thematic flexibility are certainly not.

The paper has touched on a few possible and actual extensions of the DISPLAYS ARE VIEWS metaphor as well as on combinations with other metaphors. While more should be said about these and other examples could be given, the point is that they all rely on perception-oriented rather than mapping-oriented metaphors. The fact that mapping itself is based on perception is not enough. GIS users need powerful dynamic control over what they perceive, rather than being presented with more or less static results of what a designer thought they want to perceive.

Finally, in order to become applicable to interface design, image-schematic analyses of interface metaphors like the one presented for the visual field need to be formalized. An approach based on algebraic specifications has been proposed in [Kuhn 1991]. At the same time, prototypes of interface tools which implement and visualize the metaphors have to be developed, like the ones proposed by Jackson [1990a] for zooming and panning in displays of images and maps. Current research addresses the question of how these tools can be extended to deal with zoom and pan operations in the conceptual domain.

Acknowledgements

Parts of this research have been supported by NSF under NCGIA grant No. SES 88-10917, by Intergraph Corporation, and by Digital Equipment Corporation under grant No. TP-765536. The author is also grateful for discussions about the topic with Andrew Frank, Max Egenhofer, and Jeffrey Jackson.

References

Apple Computer, Inc. 1987. *Human Interface Guidelines: The Apple Desktop Interface*. Reading, MA: Addison-Wesley.

Arnheim, R. 1969. *Visual Thinking*. Berkeley, CA: University of California Press.

Beard, K., and W. Mackaness. 1991. Generalization Operations and Supporting Structures. *Proceedings, Auto Carto 10.* Edited by D.M. Mark and D. White. (This volume.)

Bjørke, J.T., and R. Aasgaard. 1990. Cartographic Zoom. *Proceedings, Fourth International Symposium on Spatial Data Handling*. Edited by K. Brassel and H. Kishimoto. 1: 345-353.

Brassel, K.E., and R. Weibel. 1988. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems* 2 (3) : 229-244.

Brooks, F. 1988. Grasping Reality Through Illusion: Interactive Graphics Serving Science. *Proceedings, ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'88)*. Edited by E. Soloway, D. Frye, and S.B. Sheppard. 1-11. Reading, MA: Addison-Wesley.

Burrough, P.A. 1986. *Principles of Geographical Information Systems for Land Resources Assessment.* Edited by P. H. T. Beckett. Monographs on Soil and Resources Survey. Oxford, UK: Clarendon Press.

Buttenfield, B.P., and J. Delotto. 1989. *Multiple Representations: Report on the Specialist Meeting*. National Center for Geographic Information and Analysis; Santa Barbara, CA: Report 89-3.

Carroll, J.M., R.L. Mack, and W.A. Kellogg. 1988. Interface Metaphors and User Interface Design. In *Handbook of Human-Computer Interaction*. Edited by M. Helander. 67-85. New York, NY: North-Holland, Elsevier Science Publishers.

Chrisman, N.R. 1990. Deficiencies of sheets and tiles: building sheetless databases. *International Journal of Geographical Information Systems* 4 (2) : 157-167.

Conn, C., J. Lanier, M. Minsky, S. Fisher, and A. Druin. 1989. Virtual Environments and Interactivity: Windows to the Future. *Proceedings, SIGGRAPH '89 Panels.* Edited by R.J. Beach. 7-18. New York, NY: ACM Press.

Downs, R.M. 1981. Maps and Metaphors. *The Professional Geographer* 33 (3): 287-293.

Egenhofer, M.J. 1990. Manipulating the Graphical Representation of Query Results in Geographic Information Systems. *Proceedings, IEEE Workshop on Visual Languages*. Edited by S.-K. Chang. 119-124.

Frank, A. U. 1988. Requirements for a Database Management System for a GIS. *Photogrammetric Engineering & Remote Sensing* 54 (11): 1557-1564.

Furnas, G.W. 1986. Generalized Fisheye Views. *Proceedings, ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'86).* Edited by M. Mantei and P. Orbeton. 16-23. Reading, MA: Addison-Wesley.

Goldberg, A., and D. Robson. 1981. The Smalltalk-80 System. *BYTE* 6 (8): 14ff.

Goodchild, M.F. 1990. Spatial Information Science. *Proceedings, Fourth International Symposium on Spatial Data Handling*. Edited by K. Brassel and H. Kishimoto. 1: 3-12 (Keynote Address).

Gould, M.D. and M. McGranaghan. 1990. Metaphor in Geographic Information Systems. *Proceedings, Fourth International Symposium on Spatial Data Handling*. Edited by K. Brassel and H. Kishimoto. 1: 433-442.

Guptill, S.C. 1989. Speculations on Seamles, Scaleless, Cartographic Data Bases. *Proceedings, Auto Carto 9*. Edited by E. Anderson. 436-443.

Jackson, J.P. 1990a. Developing an Effective Human Interface for Geographical Information Systems using Metaphors. *Proceedings, ACSM/ASPRS Annual Convention*. 3: 117-125.

Jackson, J.P. 1990b. *Visualization of Metaphors for Interaction With Geographic Information Systems*. Masters of Science Thesis, University of Maine, Department of Surveying Engineering. Orono, ME.

Johnson, M. 1987. *The Body in the Mind*. Chicago, IL: The University of Chicago Press.

Kuhn, W. 1990. Editing Spatial Relations. *Proceedings, Fourth International Symposium on Spatial Data Handling*. Edited by K. Brassel and H. Kishimoto. 1: 423-432.

Kuhn, W., and A.U. Frank. 1991. A Formalization of Metaphors and Image-Schemas in User Interfaces. In *Cognitive and Linguistic Aspects of Geographic Space*. Edited by D. M. Mark and A. U. Frank. NATO ASI Series. Dordrecht, The Netherlands: Kluwer Academic Press; in print.

Kuhn, W., and M.J. Egenhofer (Eds.). 1991. *Visual Interfaces to Geometry*. Report on a Two-Day Workshop at CHI'90. National Center for Geographic Information and Analysis; Santa Barbara, CA: Report in print.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things*. Chicago, IL: The University of Chicago Press.

Lakoff, G. 1988. Cognitive Semantics. In *Meaning and Mental Representations*. Edited by U. Eco, M. Santambrogio and P. Violi. 119-154. Bloomington, IN: Indiana University Press.

Lakoff, G. 1990. The Invariance Hypothesis: is abstract reason based on image-schemas? *Cognitive Linguistics* 1 (1): 39-74.

Lakoff, G., and M. Johnson. 1980. *Metaphors We Live By*. Chicago, IL: University of Chicago Press.

Mark, D.M. 1989. Cognitive Image-Schemata for Geographic Information: Relations to User Views and GIS Interfaces. *Proceedings, GIS/LIS'89* : 551-560.

Mark, D.M., A.U. Frank, M.J. Egenhofer, S.M. Freundschuh, M. McGranaghan, and R.M. White. 1989. *Languages of Spatial Relations: Initiative Two Specialist Meeting Report*. National Center for Geographic Information and Analysis; Santa Barbara, CA. Report 89-2.

Marr, D. 1982. *Vision*. New York, NY: W.H. Freeman.

Minsky, M. 1985. *The Society of Mind*. New York, NY: Simon & Schuster.

Moellering, H. 1984. Virtual Maps, Real Maps, and Interactive Cartography. In *Models and Spatial Statistics*. Edited by Wilmott and Gaille.

Mohan, L., and R.L. Kashyap. 1988. An Object-Oriented Knowledge Representation for Spatial Information. *IEEE Transactions on Software Engineering* 14 (5): 675-681.

Negroponte, N. 1989. An Iconoclastic View Beyond the Desktop Metaphor. *International Journal of Human-Computer Interaction* 1 (1): 109-113.

Oosterom, P. van. 1991. The Reactive Tree - A Storage Structure of a Seamless, Scaleless Geographic Database. *Proceedings, Auto Carto 10.* Edited by D.M. Mark and D. White. (This volume.)

Robertson, P.K. 1988. Choosing Data Representations for the Effective Visualisation of Spatial Data. *Proceedings, Third International Symposium on Spatial Data Handling*. Edited by D.F. Marble. 243-252.

Robinson, A.H., and B.B. Petchenik. 1976. *The Nature of Maps: Essays toward Understanding Maps and Mapping.* Chicago, IL: University of Chicago Press.

Samet, H. 1989. *The Design and Analysis of Spatial Data Structures.* Reading, MA: Addison-Wesley.

Smith, D.C.S., C. Irby, R. Kimball, B. Verplank, and E. Harslam. 1982. Designing the Star User Interface. *BYTE* 7 (4): 242-282.

Tanaka, M., and T. Ichikawa. 1988. A Visual User Interface for Map Information Retrieval Based on Semantic Significance. *IEEE Transactions on Software Engineering* 14 (5): 666-670.

Tauber, M.J. 1987. On Visual Interfaces and their Conceptual Analysis. In *Visualization in Programming*. Edited by P. Gorny and M. J. Tauber. 106-123. New York, NY: Springer-Verlag.

Tobler, W.R. 1976. The Geometry of Mental Maps. In *Spatial Choice and Spatial Behavior.* Edited by R. G. Golledge and G. Rushton. 69-81. Columbus, OH: Ohio State University Press.

Tufte, E.R. 1983. *The Visual Display of Quantitative Information.* Chesire, CT: Graphics Press.

Tufte, E.R. 1990. *Envisioning Information.* Chesire, CT: Graphics Press.

Wilson, P. M. 1990. Get your Desktop Metaphor off my Drafting Table: User Interface Design for Spatial Data Handling. *Proceedings, Fourth International Symposium on Spatial Data Handling.* Edited by K. Brassel and H. Kishimoto. 1: 455-464.

Zeeman, E.C. 1962. The Topology of the Brain and Visual Perception. In *The Topology of 3-Manifolds.* Edited by M. K. Fort. 240-256. Englewood Cliffs, NJ: Prentice-Hall.

UGIX: A GIS INDEPENDENT USER INTERFACE ENVIRONMENT

J.F. Raper and M.S. Bundock
Dept. of Geography,
Birkbeck College,
7-15 Gresse St., London W1P 1PA

ABSTRACT

Work has begun on the design and specification of a Standard User Interface Environment for use with Geographic Information Systems. The work was prompted by the recognition that many of todays commercially available GIS products are firstly difficult to learn and use, secondly are difficult and time-consuming to customise, and thirdly the knowledge gained in using one product is not readily transferable or applicable to another. Consequently the aims of the research are to produce a prototype environment which is independent of the underlying GIS, provides high level analysis, design and customisation facilities, and presents the user with an adaptable, extensible, easy to learn and easy to use interface. In order to satisfy these aims, the research has the following specific objectives:

- to identify the common functional components that must be supported within a generic spatial language for GIS operations
- to define the form of a generic spatial language processor to support the functions while permitting input from a variety of sources such as voice, WIMP and command line interfaces
- to build a prototype generic user interface environment with interfaces to a number of commercially available GIS products
- to test user response to the interface.

This paper outlines the work performed to date and discusses in more detail the architecture of the user interface environment, the conceptual model proposed within the graphical user interface, the identification of a "standard" set of common GIS functions and the approach taken for interface customisation.

INTRODUCTION

Expansion of the Problem

The Use Environment The user environment is a vital element of any GIS. Long ignored as an esoteric aspect of GIS design while GIS development was driven by the need to extend functionality, the user environment is now beginning to attract its due attention. The development of the Universal Geographic Information eXecutive (UGIX) (Rhind, Raper and Green 1989) is a response to the widely expressed need to improve the usability of GIS, especially through the improvement and extension of the user interface. However, the implementation of a GIS user environment involves considerably more than the improvement of the human-computer interaction (HCI) process. Since GIS are conceptually complex and involve diverse operations ranging from data modelling to geometric transformations, improving the HCI cannot be a complete solution to the improvement of GIS use.

A number of general problems afflict many commercially available GIS which can be characterised as failings of the user environment. One of the most pervasive is the blurring of the distinction between goals, tasks and system functions in the language and process of interaction with the system. This means that the user cannot easily comprehend the structure of the user environment. In the design of UGIX the following definitions are used, and the concepts implemented in the interface:

GOAL	a user target for spatial data manipulation expressed in terms of application-specific outcomes e.g. finding which stands of trees in a forest will come to maturity in each of the next 5 years
TASK	a spatial data manipulation procedure expressed in terms of system implementable steps e.g. searching for certain spatially referenced items in a database and displaying the results in a map
FUNCTION	a low-level system operation to manipulate spatial data e.g. plotting a symbol at specified X,Y coordinates on an output device.

In this scheme tasks and functions refer to system operations, while goals apply to conceptual operations which are conceived of without reference to a computing environment.

Using this terminology, most GIS offer a command language composed of functions which are spatial tools and algorithms of various kinds. These commands are often modifiable with arguments and the complete expressions used are complex and often obscure. As part of a general movement to improve this situation a number of commercial systems have begun to offer graphical user interfaces (GUI's) built using window-icon-mouse-pointer (WIMP) techniques. However, these developments have illustrated the difficulties inherent in assigning icons or menu items to functions i.e. while the range of options available is now stated, the system structure is still no easier to understand. In particular, it is difficult to convert a goal into a task made up of the appropriate functions. Added to these implicit difficulties are the problems of overfilling the screen with icons or creating very long menus, the use of inappropriate screen metaphors and the lack of activity indicators to indicate the status of an operation to the user.

A further problem is that there are many different user languages for space in use, such as those defined by professionals working with spatial data (see examples in table 1). The table (with reference to two contrasting applications) shows how the relationship between objects in the application domain, user descriptions of space and the basic spatial data types supported by GIS products can be complex and difficult to understand.

The process by which the user links the concept and implementation can lead to confusion and to users making errors in they way they specify operations. A well designed user environment requires an interface which permits the customiser or expert user to link the appropriate user language for space to the system architecture in a way which is transparent to the end users. In other words the interface should allow the user to manipulate objects that are meaningful in terms of the application, like sub-divide a parcel rather than split a polygon.

Application Entity Types	Spatial Data Type	Comment
<u>Land surveying</u>		
Monument	Point	Fixed point located by physical mark
Centroid	Point	Centre point to which reference code is linked
Metes	Line	Boundaries of parcel defined by distance & direction
Bounds	Line	Boundaries defining position of adjoining parcels
Strip	Line	Corridor of fixed width either side of a centreline
Abuttal	Line	Boundary of a parcel on an adjoining parcel
Easement	Line/polygon	Corridor or area of land parcel set aside
Parcel	Polygon	Unit of land ownership
Aliquot	Polygon	Subdivision of parcel
Tract	Polygon	Land segregated by resurvey
<u>Mine Surveying</u>		
Face	Line	Section of mine boundary used for excavation
Entry	Line/polygon*	Centreline/section of tunnel forming access to face
Cross-cut	Line/polygon*	Centreline/section of tunnel at right angle to entry
Pillar	Polygon	Area of unexcavated material within mine
Room	Polygon*	Section of tunnel ending in face

* Polygon defined by closure of an open polygon in specified circumstances e.g across end of a tunnel

Table 1
Examples of user language for space for land and mine surveying

User environments of all forms have long lacked good visualisation tools for the spatial database data model. Ideally the user should be able to see the entity types and their interrelationships graphically expressed so that they can formulate queries more easily. Finally, the poor quality of help systems for many GIS has also become a major drawback for many use environments. Frequently the help is simply a formal statement of the command syntax and arguments, and not an explanation of its wider usage.

Customisability Commercial GIS software packages are normally designed to be fairly general purpose in nature - they are not designed for a specific well-defined application within a particular organisation. Consequently, they need to be adapted to fit the specific application and user requirements of the organisation within which they are implemented. This adaptation of the as-supplied system is termed *customisation*. The term *customisability* is used to describe the ease and extent to which a system may be customised.

The objectives of the customisation process are to provide a system for the user that supports both the data model and functionality demanded by the application requirements, that presents to that user an interface specific to the user's application, language and experience, which is uncluttered by non-required functions, icons and menus, is easy to learn and easy to use.

At present the base products delivered by GIS vendors are little more than a box of low-level spatial tools. These general purpose tools do not directly satisfy the user's functional requirements which are determined by organisational and application specific objectives. Furthermore, the tools will often have little meaning or applicability to the end user who must be educated in the language, interface and conceptual model supported by the product. The customisability of existing GIS products is poor, especially in the areas of database design and implementation, task definition and user interface design and development. The result is that effective customisation of a GIS product to satisfy corporate GIS requirements involves enormous expense and effort. This problem is so acute that it is not unusual to see organisations struggling to use a system that is uncustomised and uncustomisable with the available resources. Effectively, the application requirements are largely discarded so that the functions the GIS supports become the application.

The customisation process incorporates all the normal stages of the familiar systems development life-cycle, including planning, analysis, design, construction, implementation and operation. The analysis stage incorporates both data analysis (resulting in the development of data models) and function analysis which involves both process and event analysis. The design phase incorporates logical and physical database design, task design and user interface design. Construction refers to the actual development of the physical database, tasks and user interfaces, while implementation is concerned with delivering the working system to the user environment.

Non-Transferability of Skills. Each GIS product on the market today incorporates its own distinctive environment, being substantially different from virtually all other available products. Each system tends to have its own unique command language, icon set, menu organisation and form layouts. The methods of interaction with the system vary considerably, even for such simple actions as selecting an object, obtaining help information or indicating confirmation of an action. Each vendor tends to use their own set of jargon, often in a manner which is inconsistent with other GIS vendors.

Even worse, the underlying system architectures show through and must be understood by the user before effective system usage is possible. In the absence of any other strong conceptual model for the system (as might be presented in a fully customised environment), the underlying architecture (files, layers, coverages, points, lines and polygons) forms the basis of the mental model developed by the user. The application problem (e.g. forest resource management) becomes mapped to the problem of manipulating the components of the GIS architecture (i.e., the coverages, polygons etc.). Consequently the skill set acquired by a user is specific to the jargon and architecture of a particular product. Since each GIS uses different jargon and different architectures, the user's knowledge of one system is not readily transferable to another.

<u>Expansion of the Objectives</u>

Identification of Common Functional Components The first phase of the research involved identification of a set of common (generic) functions that should be supported within the UGIX interface. These functions must be independent of any underlying GIS implementation strategy (e.g. object, relational or sheet based) and dependent rather on the goals of the user. These functions will be accessed via a set of icons, menus and forms within the GUI, and as a set of commands, operators and procedures within the command language. A "standard" set of icons and command names will be provided for the generic functions which may be modified (when required) within the customised environment. It should be noted that the generic functionality will not be implemented within UGIX, but rather, it will be accessible through UGIX. This distinction is important, since it restates the concept of separating the application from the user interface.

By identifying the functions required to satisfy tasks, and tasks to satisfy generic goals, and providing access to these via icons and commands, the interface should become more consistent and less dependent on the underlying GIS data structures and architecture. It is believed that this will make the system easier to learn and use, and once learned will provide the user with knowledge that should be more readily transferable to other systems.

Definition of a Generic Spatial Language Processor A command language for interaction with the GIS database that supports the generic functions is being developed. It is intended that the spatial language will be embedded within both 3GL and 4GL languages which will provide the program logic and control structures. A spatially extended form of SQL (SQL-SX) has been designed to provide a standard, transportable language suitable for database definition, query, insertion, deletion and update. SQL-SX is to be supplemented by the set of generic GIS functions identified above. The overall architecture for the spatial query processor has now been sketched out. It includes layers for SQL-SX, an equivalent iconic query language, an inter-process communications interface and a customisation environment.

Development of a Prototype Generic Use Environment To test the feasibility of the concepts described here and the usability of such an interface, a prototype use environment must be developed. Further, it must be interfaced to a number of commercially available GIS products to prove that the user interface can be detached from the underlying GIS product architectures. The more difficult aspects of the development are likely to be:

- creating an efficient system to map between the functions supported within the generic spatial language and the matching functions supported by the underlying GIS,
- hiding the user interface supplied with the underlying GIS.

Testing User Response The resulting prototype must be evaluated to determine that it does indeed provide a superior user interface environment to the standard interfaces provided by each of the underlying GIS products. To test the acceptability of the UGIX environment, a number of trials are proposed. Methods for evaluation of user interfaces include:

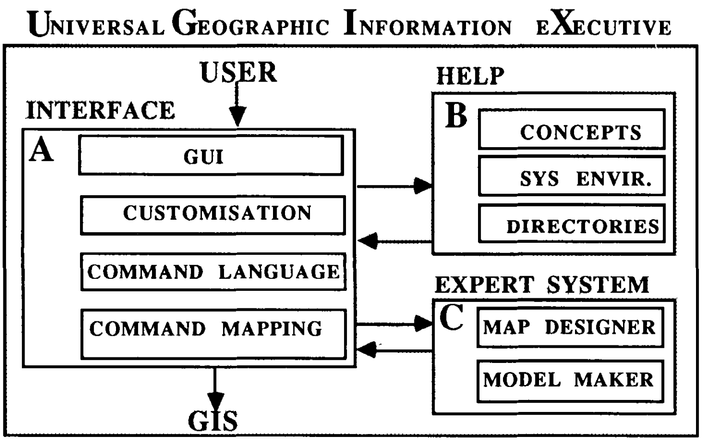
- formal analytical methods where the interface is evaluated in isolation from users (Grudin 1989)
- empirical methods where users are requested to perform the same tasks using different interfaces, and the performances measured and compared
- ethnographic methods where actual users are observed and the context and users observations are elicited (Crellin, Horn and Preece 1990).

During design of the graphical user interface we propose to explicitly adopt accepted guidelines (e.g. proposed ISO guidelines Williams 1989, Strijland 1990) as an aid to user interface specification. Analytical methods, operating on data gathered by automated monitoring of user interaction, may be used subsequently to determine interface effectiveness, identify frequent command usages, common errors and the relationships between use patterns and error occurrences. Chen (1990) describes the use of monitoring facilities built into the Xt toolkit to automatically gather appropriate information. Empirical studies are also proposed to compare the effectiveness of the new environment in direct comparison to the interface provided by the underlying GIS. Finally, user acceptability will be evaluated based on user interviews. It is intended that this evaluation will lead to suggestions for improvement for subsequent developments.

Background to UGIX

System Overview The UGIX system design as described by Rhind, Raper and Green (1989) contains 3 main modules, viz. (A) containing the screen interfaces, dialogues and command processor; (B) containing a help and information system for a GIS; and (C) an expert system shell or high level system access module. The structure of UGIX is illustrated in figure 1. This section describes the approach taken in the UGIX project, through first and second generation implementations. The major distinction between the generations is that the first aims to improve the usability of a specific GIS implementation, while the second aims to provide a generic user environment supporting transfer of skills between GIS and allowing easier customisation.

Figure 1. The three primary modules of the UGIX architecture.



The first generation approach to interface design within the UGIX project has been to prototype using Hypercard for the Apple Macintosh, where the Hypercard application (complete with in-built communications software) acts as a client to a host processor running the GIS application software (Raper, Linsey and Connolly 1990). This approach is similar to the one used by Cowen and Love (1988) to create an interface to the South Carolina Historic Preservation Office GIS database. Hypercard with its standard set of buttons, scrolling boxes and cards makes use of the GIS less daunting for the less technically-aware user. In addition, with the rich graphics environment available in Hypercard it is possible to show a graphic to illustrate the effect of various options available at any one point. It is also desirable to display all the commands available to the user in one place, with a pop-up explanation for each option.

Screen design has involved the standardisation of button and text field formats as well as card and background layout for different areas of activity such as:-

- Introduction and explanation (using a map guide);
- Map and file selection (using standard Macintosh file selection dialogue);
- Session screens for command processing;
- Help environment (UGIX (B) based on GISTutor version 2);
- A Gallery for maps and images generated in the GIS (along with button to redraw them).

Screen metaphors have been developed for each of these areas to make location in the system a graphical attribute. The interface also displays an activity index to give continuous feedback to the user on the status of the session. Currently this system interface 'shell' is being implemented for the GIS ARC/INFO, and is known as 'HyperArc': it is currently under test with users at 'novice' and 'competent' levels of expertise. In addition to feedback on the use of the system, the aim of the evaluation phase is to define a core area of functionality in common use to help optimise the UGIX system structure.

An important early objective in the development of HyperArc was the creation of file handling procedures to harmonise the user's concept of maps with that of ARC/INFO. This establishes that maps are both 'views' of spatial data and sheets within a series i.e., spatial tiles. Thus, search routines to find maps with particular names, to sort maps by type (e.g. point or polygon based), to access the map tiling system and to select the part of the sheet to view have all been created. In the first generation of the UGIX project the user specifies spatial queries using these system implementation concepts which are made comprehensible to the user diagrammatically (user testing is helping to refine this aspect). Hence HyperArc forces the user to work with ARC/INFO concepts, but tries to connect them with the user's view of the problem under study. This is ultimately restrictive to the user since the data structure is fixed, and maps are files which the user needs to manipulate in some way.

A basic principle of the UGIX design is that in order to make a GIS easy to use the process of making a database selection, displaying a map or carrying out spatial analysis must be broken down into a series of logical parts, linked by a pathway for the user to follow. Following such a path and gaining experience with the alternative options is an excellent way to improve a user's end-to-end understanding of the components of spatial data processing. Appropriate information needed for a user to make a decision is also retrieved before indicating the command options, for example only maps with the correct specifications are presented (e.g. with topological relations already created), when this is necessary for the operation.

Another UGIX design principle is that improving access to existing GIS can be achieved by converting the current function-orientation of the native system interface (primitive and implementation-specific operations) to a task-oriented interface (sequences of high level spatial operations) usable by a spatially aware user. The second generation of the UGIX project aims to build on the experience of constructing such task-oriented interfaces to create generic interfaces capable of communication with any GIS.

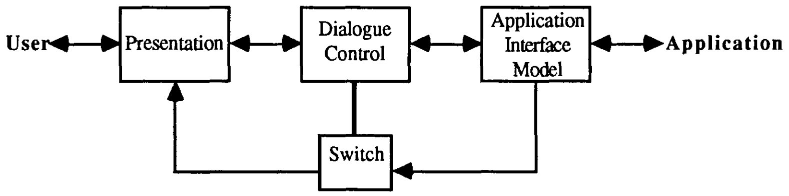
However, in order to implement such an interface in a generic way requires a new form of software architecture which is independent of specific implementations, does not enforce a particular data model, and adheres to the standards in the user community which are most crucial to the success of GIS in heterogeneous computing environments. To achieve this a layered model is suggested that protects the user interface from the actual implementation mechanisms provided by each GIS vendor. Each layer within the model will perform a particular task and have a well defined interface to the layers both above and below. Some of the layers within UGIX will be able to communicate directly with the underlying GIS at a matching level.

UGIX(A)

Design Overview

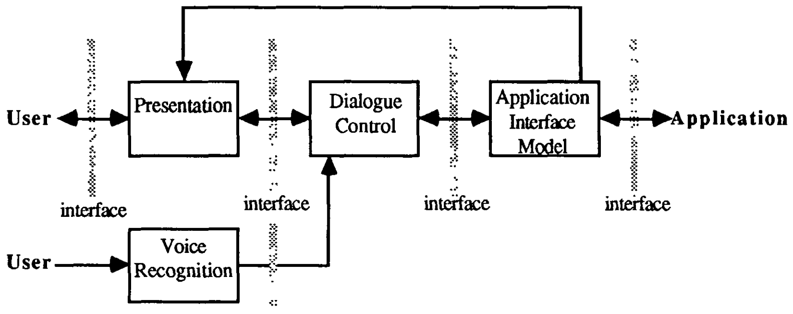
Separation of the user interface from the application is not a new concept. Early work resulted in what is often known as the "Seeheim model" (Green 1985) developed during a 1983 workshop on architectures for interactive systems at Seeheim. Subsequently, the identification of components of the overall system corresponding to semantic, syntactic and lexical aspects, and the relationships between them has lead to various alternative architectures. The development of general purpose software for managing the user interface as a separate process has lead to the comcept of the user interface management system (UIMS) (Pfaff 1985). Here the application and the user interface software are quite separate and communicate via a well-defined protocol.

Figure 2. The Seeheim model



The overall architecture for UGIX(A) is similar to the Seeheim model in many aspects. The requirement for a high bandwidth communications channel from the GIS application is supported to allow efficient graphics display and manipulation. Figure 3 illustrates the overall structure proposed for the UGIX environment.

Figure 3. Overview of UGIX(A) architecture



Description of Components

The presentation layer incorporates a standard user interface toolkit (e.g. Motif, OpenLook etc.) a widget design facility, a screen design facility and a screen execution facility. The widget and screen design facilities operate within the constraints of the toolkit, and will be implemented as a set of executable screens. The customisation environment itself and the actual resulting end-user application, will also simply be a set of screens with which the user may interact.

The screens will be designed in terms of a set of windows, a set of widgets within each window and a set of forms. The behaviour of the windows, widgets and forms will be described in terms of the 4GL command language. Interaction with the screens will cause the widgets to react in the predefined manner and the execution of GIS tasks in terms of the spatial command language embedded within the 4GL.

Equivalent commands may be issued directly via a command line interface, via widget interaction or using voice input. Each method should result in the execution of the same generic functions within the dialogue control component. The voice recognition facility issues either individual spatial command language tokens which may be used to build a complete command, or entire commands. Entire commands may be abbreviated into a spoken shorthand consisting of just a few words rather than requiring the user to speak the full command syntax for a particular task.

The application interface module accepts generic spatial language commands and maps them onto the command language of the underlying GIS. It then issues these commands to the GIS via an inter-process communications mechanism. The GIS responses may include alphanumeric information (which may be used to fill a form), status information (error and function status) and/or graphics. To support a highly interactive graphics environment requires that a high speed channel be provided to display the graphical data. However, alphanumeric and status data may be routed through the dialogue control module for further processing and display.

Figure 4 illustrates the proposed architecture of UGIX(A) in more detail.

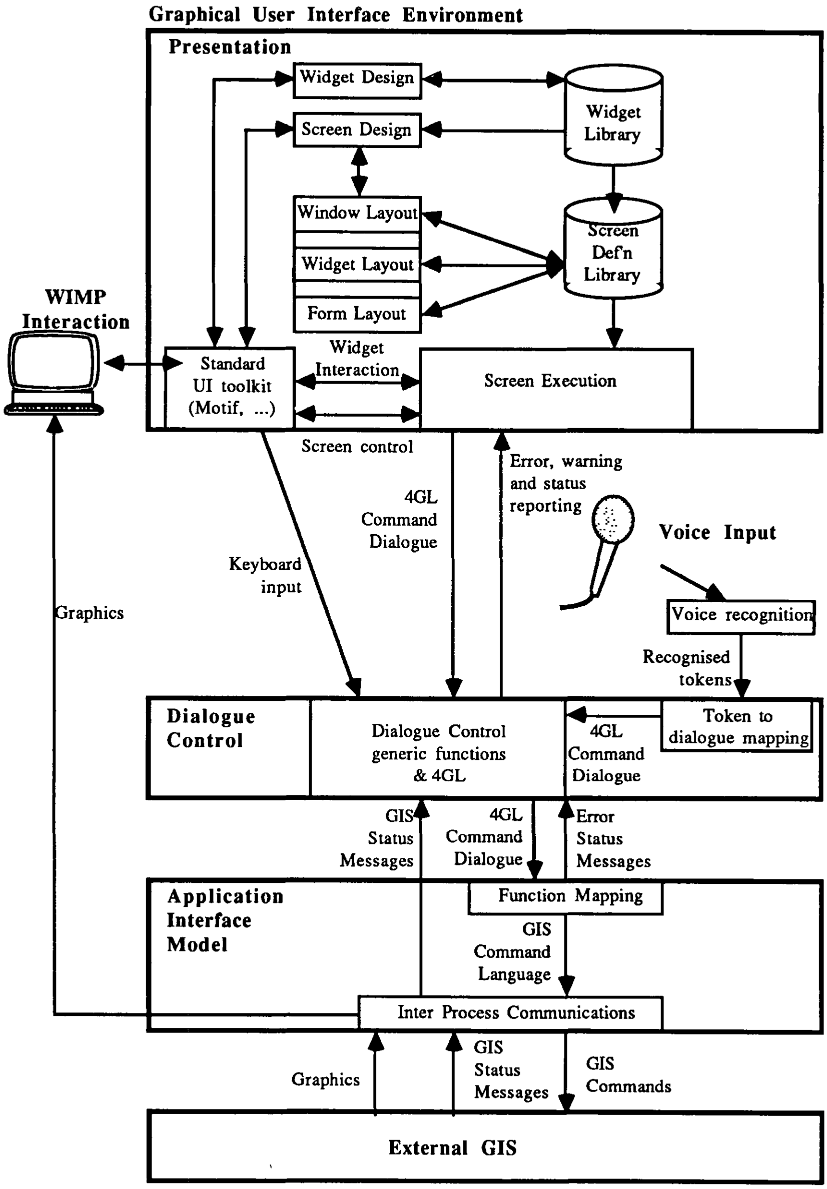
The Graphical User Interface

Wilson (1990) reviews the use of graphical user interfaces within GIS and the applicability of the desktop metaphor. He suggests guidelines for building suitable user interfaces. The GUI is described as having three components:

- an underlying conceptual model,
- a command structure comprising codes, function keys, buttons etc. with which to create a syntax, and
- the visible screen graphics, such as command lines, menus and icons.

To date, within the GIS world, most emphasis has been placed on the development of a command syntax and the design of menus, icons and screens. However, as yet there are no agreed standards for interaction with the interface unlike the PC world where techniques such as double clicking on an object to activate it, F1 function key for help etc. are commonly adopted.

Figure 4. More detailed breakdown of UGIX architecture



This lack of standardisation has lead to a lack of transferability of knowledge, long learning periods and generally difficult system usage.

Initial attempts at improving usability concentrated on reducing the number of interactions, menus, forms and icons that the user had to deal with. This approach either reduced the available functionality or produced menu hierarchies that were difficult to use. An alternative approach is to use existing knowledge of a related field that may be applied to the new problem domain.

The Underlying Conceptual Model A major contributing factor towards the non-standardisation of the GIS user interface is the lack of an underlying conceptual model for the interface. It has been suggested (Gould & McGranaghan 1990) that the primary mechanism by which a user learns to use GIS is by metaphoric learning. Here the user is able to treat the unfamiliar environment like another familiar one thus reducing the overall learning period. The general cognitive process may be partitioned into metaphoric, analogical and modelling processes. The differences between the three processes and their implications for computer systems design are reviewed by Wozny (1989). The concepts of metaphor and analogy are closely related: analogy implies that one domain behaves like another, whereas with metaphor, the target domain is more directly mapped onto the other and hence becomes the other. Consequently, the use of metaphor within the user interface is preferable since it allows a user to interact with an unfamiliar system as if it ***is*** an environment with which they are familiar. This effectively reduces the learning time, reduces stress caused by unfamiliarity (i.e. makes for a happy user) and provides a conceptual framework for the new environment which may be built upon. For infrequent users, the use of metaphor may be more important, since they may never progress beyond the metaphor presented to develop a mental model of there own (Wozny 1989).

Existing graphical user interfaces for non-GIS applications have often been developed using the desktop metaphor as the underlying conceptual model. The desktop metaphor is suitable for many business related applications since the activities performed by the computer based application have direct equivalents with the manual methods. However, it may not be readily applicable to many GIS applications due to the lack of spatial and mapping related activities that normally occur on and around a desk.

The wide variation in GIS applications and the variation in experience of GIS users indicates that a single conceptual model is unlikely to satisfy or be applicable to all situations. If we perceive GIS to be an enabling technology for the integration of spatial and aspatial data, we must then consider it to be equivalent to a DBMS in generality, and hence not suited to a single model. In contrast, a GIS customised to suit a particular narrow application (e.g. mains fault analysis in the Water industry) may provide a situation where an applicable underlying conceptual model may be utilised.

Wilson (1990) pointed out that some GIS applications may have no equivalent manual method. However, this does not imply that a conceptual model on which to base the user interface cannot be found. Rather it implies that analogy or metaphor may be suitable techniques for development of the conceptual model.

Current GIS technology imposes on the user a conceptual model of geographic space that is a function of the internal structures supported by the GIS (e.g. layers, points, lines, polygons). What we should be aiming for is a user interface that permits the system customiser to present a conceptual model to the user that is relevant and applicable to the both the user's background and the application in hand.

The strength of the desktop metaphor as used within the Macintosh and other PC environments for the underlying conceptual model, is that it provides an organising framework within which other operations and metaphors may exist. Gould and McGranaghan (1990) have extended this idea to suggest the need for an organising metaphor within which there may be other nested metaphors (which may themselves be organising metaphors). This approach has promise since it provides a structure within which applicable and relevant metaphors may be applied, rather than trying to apply a single metaphor to all situations.

The Organising Metaphor within UGIX The current thinking for the UGIX GUI is to develop an environment supporting nested metaphors. The proposed overall organising metaphor is a building, within which there are a set of rooms, each accessible via a door. It should be noted that the idea of using the room/building metaphor has been independently conceived by a number of different groups including researchers at Xerox Palo Alto Research Center and University of Waterloo (Chan and Malcolm, 1984), and even built into a number of existing products (e.g. Rooms from ENVOS and even X11 rev 4 attempts to provide a Rooms-like system).

Within UGIX, each room may possess its own organising metaphor. Most rooms will be directly accessible from the entrance hall although some special-purpose rooms may require access from within another room. On entering the system the user is located in the entrance hall, a neutral, public space through which the user moves to particular environment. Doors provide access to the environments the user has access to. The door metaphor is a strong metaphor for access into and out of different environments (Cátedra 1990) and provides features such as locks, opening and closing. These features may be used directly for access control, entering and leaving. Within each room, a single type of activity occurs, and a single lower-level organising metaphor is employed.

For example, one room provides an environment where the desktop metaphor is supported. Here general filing, correspondence and interfacing to external non-GIS packages (e.g. word processing, spreadsheets) takes place. Access to the aspatial part of the GIS database is available via card indexes, file folders etc. and alphanumeric reports can be created and printed.

A second room contains the drafting office where the map, drafting table, map cabinet and light table are the principle metaphors. Now access to the GIS database is via the map. Maps may be taken out of the filing cabinet, updated, viewed and copies taken for development proposals etc. Eventually these may be replaced in the map cabinet following approval by the chief draftsman/engineer. Note that operations not consistent with the map metaphor may not be applied here.

A third room contains a library. Within the library books are kept providing reference material, reports, archives and documentation. Updating of system documentation is performed here.

A fourth room contains the development and customisation environment. A workstation metaphor •is supported which provides direct access to the 4GL development and customisation environment.

There is one special door that leads off the entrance hall - an external door. Through the door, blue sky and clouds may be glimpsed, and on entering this environment the real world metaphor is used for access to the GIS database. Here there are no seams, maps or files - only a continuous world containing objects. This is where the experienced GIS user works and it is also the environment in which virtual reality may one-day be accessible (Jacobson 1990).

Users are provided with keys that are able to unlock only some doors. It is feasible to consider more specialised rooms leading off of others. For example the database administrator and system manager may be in their very own room accessible from the development and customisation area.

The concepts of buildings, rooms and doors are internationally and culturally neutral, providing an almost universally understood concept. A further advantage exhibited by this metaphor is that it provides an easy facility for extension. To add new environments involves simply adding another room (with door!) to the building. Within each new environment a different organising metaphor may be used to support functions not supported elsewhere. The possibilities of this metaphor are seemingly endless - e.g. leaving the system may simply be performed by turning off the light switch in the entrance hall, or alternatively going through the door that leads out into the night.

The Iconic Query Language Access to the functionality provided within each environment (room) will be predominantly via icons, menus and forms. The icons should fit the organising metaphor for that environment so that they have relevance and preferably direct applicability. Consequently, the drafting office might be designed specifically for experienced cartographers and hence might support map cabinets from which maps may be extracted, drafting boards on which map updates and viewing may be performed and light tables on which map overlay operations may be carried out.

Consistency and simplicity are key considerations when attempting to design a user interface, be it for a GIS or for a dishwasher. A concise and simple syntax for manipulating the icons and database objects is required which is both consistent and meaningful in terms of the metaphors used. Existing GUIs such as that used on the Macintosh are not fully consistent. Consistency between applications has been encouraged since Apple provide a set of guidelines for developers to follow (ref Apple Mac developers guide). Consequently most packages available for the Mac have a similar look and feel so that knowledge of one application/package provides useful knowledge on the use of others.

Certain other aspects of the Mac interface are far less consistent. In particular the order in which the objects and the operators are selected varies from one type of operation to another. Objects to be manipulated are usually selected first, and then the operation to be applied is selected (e.g. discarding data by moving it to the wastebasket, applying a different font and ruler to a section of text). However, sometimes the operation to be performed is selected first and then the objects to which it is to be applied are identified (e.g. select the print function and then indicate which pages to print). Hence, even though operations that are common between applications are normally presented in a very similar manner the syntax for different operations may vary within an application.

A further level of inconsistency, most frequently observed by novices, is the use of the wastebasket. Why is the trash can used to eject the disk when it is normally used for deleting data? Most novices are unable to find a method for ejecting the disk, since the use of the wastebasket for deleting documents and folders implies that if the diskette is moved to the wastebasket, all folders and documents on the disk will be deleted. This latter example indicates where the use of a metaphor has been extended beyond its applicability and used in an inconsistent manner.

The impact of the English language on the syntactic ordering of operations, parameters and objects (verbs, modifiers and nouns) may not have relevance to the iconic interface. Although it is known that language structures our concept of space (Talmy 1990), it is not thought that language will adversely impact the syntactic structure of the iconic interface. In English we generally use a noun-verb-modifier ordering to state facts (e.g. Jack closed the door), but a verb-noun-modifier ordering for instructions or commands (e.g. close the door quietly please). Most of the operations performed within the GIS tend to be instructional in that the user is commanding the system to perform some action (e.g. modifying, deleting, reporting), supporting the adoption of a verb-noun-modifier ordering.

However, most iconic interfaces require that the objects are selected prior to identification of the action (i.e. noun-verb-modifier, or object-action ordering). Even though this ordering is not common within the English language for instructional sentences, it does feel natural for English speaking users of the iconic interface.

Perhaps the most important aspect of this ordering is that object selection and the operations to be performed on those objects are effectively separated. They have become two discrete instructions issued by the user. Furthermore, object selection is common to virtually all operations and becomes independent of those operations, meaning that a single set of object selection techniques can be applied throughout.

One significant disadvantage to the use of object-action syntax ordering is that the selection process may select objects for which the operation may be invalid. If the operation to be performed is identified first, the object selection process can use knowledge of the operation to ensure that only appropriate (valid) objects are selected. Within existing GUIs, this problem is partially overcome by disabling functions for which the selected data is inappropriate. However, if it is not obvious which of the selected objects is causing the function to become unselectable, much operator frustration is likely to ensue.

Within the UGIX GUI, we recommend the use of object-action ordering as the basic syntactical construct for icon interaction. Object selection will be performed first. Subsequent identification of an action will apply that action to the selected set of objects.

The Command Level Interface

The command level interface incorporates a 4GL command language, a function mapping facility and an inter-process communications facility. It accepts commands from the GUI and the voice recognition system in terms of 4GL command sequences. It can also be accessed directly to perform ad hoc functions and applications development.

Smallworld Magik, an object-oriented (OO) development facility from Smallworld Systems has been selected for the development of the prototype.

Smallworld Magik: An object-oriented development environment. The objective is to support a single command line and development environment in which application development, database definition, ad hoc queries, menu, form and icon commands, database access and graphics are all available. It is also desirable that the full power of the underlying GIS, DBMS and UIMS are available. An object-oriented development language has been selected since it offers the opportunity for high programmer productivity and a structured development approach. The use of OO techniques such as code re-usability, inheritance and encapsulation can reduce the overall development effort for a complex system. The Magik language (Chance, Newell and Theriault 1990) is a hybrid of the procedural and OO approaches and supports its own interactive development environment. It is fairly readable (certainly more so than 3GLs such as C and C++), comes with a comprehensive set of standard object classes, methods and procedures, and provides the ability to transfer applications between hardware and operating system platforms with a minimum of effort. It utilises the X-Windows standard for all interaction with the workstation.

For the UGIX development, we are extending Magik by adding a new set of language constructs to support a spatially extended version of SQL.

SQL-SX: A spatially Extended version of SQL The relational language SQL (Date 1989). forms a suitable base on which to develop spatial extensions due to:

- its widespread acceptance by database users
- its availability within a large number of commercially available DBMS (relational and non-relational)
- its acceptance as an international standard
- its ongoing development, thus ensuring a long-term future.

The use of relational database management system (RDBMS) technology within the existing GIS user community is virtually universal. Further, it is likely that RDBMS will be the major data management technology for at least the 1990s, and that SQL will be the major language for interaction with those databases. The investment by user organisations in training staff in the use of SQL is significant so there is consequently a sizeable body of expertise available both within GIS user organisations and in the general computing industry. The use of a non-standard query language for GIS implementations does not appear commercially viable, nor practical in the near future.

A number of GIS vendors are already developing spatially extended versions of SQL and have reported their work in the research literature including Kork (Ingram & Phillips, 1987), Intergraph (Herring, Larsen & Shivakumar, 1988), Wild Heerbrugg (now Prime) (Charlwood, Moon & Tulip, 1987) and GeoVision (Westwood, 1989). Each has attempted to provide facilities supporting spatial predicates and spatial data manipulation facilities within SQL (or SQL-like query languages). Unfortunately, the basic query language in each case has been an incomplete implementation of the ANSI/ISO SQL standard (ISO 1987, ISO 1989), and the spatial extensions were fairly minimal and elementary. To make matters worse, the extensions in general do not maintain consistent syntactic and semantic constructs with the rest of SQL. For example, spatial predicates are not in general supported within the WHERE clause, but rather within a separate clause.

Other researchers including Pong-Chai Goh (1989), Sacks-Davis, McDonell and Ooi (1987) and Egenhofer (1987) have also provided useful contributions towards the development of spatial extensions to SQL. However, until recently there has been no proposal for a standard set of extensions put forward for discussion. In our recent paper (Raper and Bundock 1990) we proposed a set of spatial extensions for SQL that could form the basis for an agreed standard between GIS vendors and the GIS user community. The spatial extensions are based on the existing proposals for SQL2 and SQL3 being studied by the combined ISO-ANSI SQL standards working group (ISO-ANSI 1989). These proposals include a number of object-oriented concepts, including support for abstract data types, methods, operators and functions. In particular, the detailed proposal in support of abstract data types (Kerridge 1989) if implemented, would provide the framework in which to develop spatial data types, spatial operators and spatial functions, while remaining completely within the SQL standard.

The extensions necessary to make SQL usable within GIS applications for query of both spatial and aspatial data include:

- spatial data type(s) e.g. point, node, line, polygon
- spatial operators (predicates) e.g. at, inside, encloses, connected_to
- spatial functions e.g. area_of, length_of
- long transaction management statements
- report specification facilities - both textual and graphical

In addition, functional requirements demand that:

- the spatial data types should be displayed graphically as a result of being SELECTed
- the data dictionary and DDL support the extensions
- spatial access control (protection) may be provided by inclusion of spatial data types, predicates and functions within VIEW definitions
- spatial integrity maintenance may be provided by support of spatial data types, predicates and functions within the CONSTRAINT clause.

The Function Mapping Facility. This facility accepts UGIX commands in the form of function specifications and data selection specifications, and transforms these to the command language of the underlying GIS. For functions supported by both UGIX and the underlying GIS, the mapping should normally be moderately straightforward. Using OO techniques, the function mappings become methods for the function objects. It is hoped that this technique will provide a straightforward approach to allow the support of multiple underlying GIS products.

Functions in UGIX not supported by the GIS However, not all functions known to UGIX may be available in the underlying GIS. Consequently, there will be holes in the UGIX interface for those unsupported functions. Icons representing unsupported functionality will be displayed in grey, and the matching command level functions will possess a method that reports on the function unavailablity.

Functions in the GIS not supported by UGIX Where the underlying GIS supports functions unknown to UGIX, a general purpose facility will be provided that allows UGIX to send the system dependent command string directly to the underlying GIS. The command string will be explicitly declared and may be associated with icons, menus and forms in the normal way. In this way, special purpose functionality may be readily included in the UGIX GUI, even if it is not considered generic enough to warrant inclusion in the generic function list.

SYSTEM CUSTOMISABILITY

The Requirement for Customisation

Customisation of any GIS is required to allow the system to manage and manipulate the entity types that exist in the problem domain. The system customiser or database administrator, must be able to describe to the system the object classes/entity types that are to be modelled within the resulting system. They must define the names of the object classes, the names and types of attributes the objects possess, the behaviour of the objects with respect to operations on the objects and the inter-relationships between objects. The object class names, attribute names and operation names should each be assigned in terms of the language normally used within the application (i.e. the application specific jargon - e.g. the land surveying examples of table 1) not in terms of the language used by the GIS. Defining the objects, attributes and operations in terms of the user's language allows interaction with the system to use that language.

The user interface may be customised to reflect the names described above and the symbols used by the application. Consequently, the standard icons representing the object classes, attribute types and operations of the application domain must be generated and associated with the names of the matching elements. This permits interaction with the GUI to be performed with icons recognised by the user as being part of the application domain.

The graphical user interface may also be customised to provide a conceptual model using metaphors which inexperienced users may recognise from previous experience - either application experience or experience from other domains. This conceptual model must be appropriate to both the application and the user's background.

The customisation facilities must be continually available, rather than being used just once to create a fixed, static system. Business (research, education, ...) requirements change, resulting in either changes to existing applications or entire new applications being created. The user interface must also remain adaptable to match individual user preferences, and user-specific tasks.

Integration of Task Analysis Methodologies

A variety of methodologies and associated tools are available today to assist system designers and customisers determine the requirements for a new information system. User-centered requirements analysis methodologies provide a structured approach to performing both data and task analyses.

Data analysis results in a detailed description of the object classes, their attributes and the inter-relationships between object classes that might be managed within the final system. It also identifies integrity constraints and any other special behaviour exhibited by objects when a change of state occurs. Although this information may be recorded on paper in a descriptive manner, it is also possible to save it in a database - often refered to as the data dictionary. Tools that assist the data analysis task will nearly always save this meta-data in a form that may be used at a later date for automating the creation of the target database.

Task analysis results in the development of a detailed description of the goals, processes and user interaction that must occur for the goals to be met. Tools that assist this process may also store the task descriptions in a database in a structured form. Formal mechanisms have been developed to structure this information in such a manner that it may be used later for automating the creation of the user interface. For example Extended Task Action Grammar (ETAG) (Tauber 1990) may provide such a mechanism, although it is likely that the level of detail required to define the target system may be significant.

It is intended that UGIX(A) incorporate tools to assist both data and task analysis. These tools will be used to gather and structure information describing the target environment in such a manner that it may subsequently be used to automatically generate the database definition and the user interface. Further, the information should be in a form to provide input to the help facility supported by UGIX(B) since descriptions of the data, the low level functions and the tasks will all be available in a structured form.

COMMON FUNCTIONAL GIS OPERATIONS

A Methodology for Identification of Key GIS Functions

Analysis of the functions to be supported directly within the interface is based on goal analysis and hierarchical decomposition (top-down) techniques rather than task analysis techniques since the interface, as delivered, must retain its generality. Task analysis might provide a more detailed definition of requirements for a particular (well defined) application but is considered too specific when attempting to define general requirements. Abstraction of the general functionality from a task based analysis might be possible given sufficient access to a wide range of actual users in a wide range of applications. However, operating within a tight set of time and resource constraints, requires the analysis to be performed quickly with minimal contact with real users.

To overcome this limitation, the study initially involved :

- a review of existing research literature,
- a review of a number of GIS tenders,
- a review of the functionality supported by a number of commercially available GIS products.

A number of authors have reported attempts at classifying GIS functionality, including Dangermond (1983) and Maguire and Raper (1990). However, Dangermond based his analysis firmly in terms of the map concept which partitions the world into discrete tiles. Consequently, much of the functionality described is concerned with managing these discrete units, to allow queries across multiple map sheets, to join multiple maps to form a new composite map, and to perform edge matching. This functionality is quite distinct from the goals of the user, being the functionality necessary to support a map-sheet based GIS implementation. Maguire and Raper describe the functional components of a GIS in a more generic manner. Below each identified high level function they separate the functionality that applies to the spatial data, from that which applies to the attribute (aspatial) data. In the current study we attempt to retain the user's concept of objects within their application domain which may possess spatial and/or aspatial components. This suggested set of generic functions will be presented with the first UGIX prototype.

CONCLUSIONS

The development of a GIS independent user interface environment capable of interfacing with a number of commercially available products while still providing an adaptable, consistent, easily learned and easy to use interface, appears at first sight a difficult task. A structured approach to this problem is however beginning to indicate the feasibility of the project.

Incorporating structured analysis tools into the environment is expected to simplify system customisation while improving the resultant interface. In particular, it should help provide a user interface that incorporates the terminology, icons and conceptual models specific to the application and user's background while developing the interface in terms of a set of standard guidelines.

The usability of the prototype system will be evaluated and compared to the interfaces offered by the underlying GIS products. The comparison will eventually prove or disprove the viability of this approach. If successful, we would like to promote the concepts incorporated within UGIX to initiate discussion between users and vendors on the development of standards and guidelines for GIS user interface design and construction.

ACKNOWLEDGEMENTS.

The authors of this paper acknowledge the help of ESRI in the funding of work leading to the prototyping of HyperArc, the first generation of UGIX. Apple Computer (UK) have also assisted in this work by establishing a laboratory of Macintosh computers which form the Apple Mapping Centre at Birkbeck College. Receipt of an ESRC/NERC grant from the collaborative programme in GIS is acknowledged by MSB. Smallworld Systems have assisted the project with hardware and software tools. We would also like to thank Lesley Bundock for her efforts in reviewing and commenting on various drafts of this paper.

REFERENCES

Apple Computer (1987). *Human interface guidelines.* Addison Wesley, Amsterdam

Cátedra M. (1990). "Through the Door": A view of space from an anthropological perspective. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space,* Las Navas, Spain

Chan P.P. and Malcolm M.A. (1984). Learning considerations in the Waterloo port user interface. *Proceedings IEEE First International Conference on Office Automation,* IEEE, New York

Chance A., Newell R.G. and Theriault D.G. (1990). An Overview of Magik, Technical Paper 9/1, Smallworld Systems Ltd, Cambridge, UK.

Charlwood, G. Moon, G. and Tulip, J. (1987). Developing a DBMS for Geographic Information: A Review, *Proceedings Auto-Carto 8*, Baltimore, Maryland.

Chen J. (1990). Providing Intrinsic Support for User Interface Monitoring, *Proceedings Human-Computer Interaction - INTERACT '90*, Cambridge, UK.

Cowen D.J. and Love S.R. (1988). A Hypercard based workstation for a distributed GIS network. *Proceedings GIS/LIS '88*, San Antonio, TX, USA.

Crellin J., Horn T. and Preece J. (1990). Evaluating evaluation: A case study of the use of novel and conventional evaluation techniques in a small company. *Proceedings Human-Computer Interaction - INTERACT '90*, Cambridge, UK.

Dangermond J. (1983). A Classification of Software Components Commonly used in Geographic Information Systems. *Proceedings United States/Australia Workshop on Design and Implementation of Computer-Based Geographic Information Systems*, Amherst, New York

Date C.J. (1989). *A Guide to the SQL Standard*, Second Edition, Addison-Wesley, Reading, Massachusetts

Egenhofer M.J. (1987). An extended SQL syntax to treat spatial objects, *Proceedings 2nd International Seminar on Trends and Concerns of Spatial Sciences*, New Brunswick.

Green M. (1985). Report on Dialogue Specification Tools. In Pfaff, G.E (ed) *User Interface Management Systems*. Springer Verlag. pp 9-20.

Gould M.D. and McGranaghan M. (1990) Metaphor in Geographic Information Systems, *Proceedings 4th International Symposium on Spatial Data Handling*, Zurich, Switzerland.

Grudin J. (1989). The Case Against User Interface Consistency, *Communications of the ACM*, 32, 10.

Herring J.R. Larsen R.C. and Shivakumar J. (1988). Extensions to the SQL Query Language to Support Spatial Analysis in a Topological Data Base, *Proceedings GIS/LIS '88.*

Ingram K.J. and Phillips W.W. (1987). Geographic Information Processing Using a SQL-Based Query Language, *Proceedings Auto-Carto 8*, Baltimore Maryland.

ISO (1987). Report ISO 9075 Information Processing Systems-Database Language SQL

ISO (1989). Report ISO 9075 Information Processing Systems-Database Language SQL

ISO-ANSI (1989). Database Language SQL2 and SQL3 (ISO-ANSI working draft) X3H2-89-252 and ISO DBL FIR-3, J Melton (editor), July 1989

Jacobson R. (1990). Virtual Worlds, Inside and Out. *Proceedings NATO ASI Cognitive and Linguistic Aspects of Space*, Las Navas, Spain

Kerridge J.M. (1989). A Proposal to add User Defined Datatypes to SQL, ISO report ISO/TC97/SC21/WG3-DBL-FIR-3.

Maguire D.J. and Raper J.F. (1990). Design Models and Functionality in GIS, *Proceedings GIS Design Models,* Leicester, UK.

Pfaff G.E. (ed) (1985). *User Interface Management Systems,* Eurographics Seminars, Springer-Verlag, Berlin

Pong-Chai Goh (1989). A GQL for Cartographic and Land Information Systems, *International Journal of Geographic Information Systems,* vol. 3, no. 3, 245-255.

Raper J.F. and Bundock M.S. (1990) UGIX: A Layer Based Model for a GIS User Interface. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space,* Las Navas, Spain

Raper J.F. and Green N.P.A. (1989). Development of a Hypertext Based Tutor for Geographical Information Systems. *British Journal of Educational Technology.*

Raper J.F., Linsey T. and Connolly T. (1990). UGIX: A Spatial Language Interface for GIS : Concept and Reality. *Proceedings of EGIS '90,* Amsterdam, The Netherlands

Rhind D.W., Raper J.F. and Green N.P.A, (1989). First UNIX then UGIX, *Proceedings of AutoCarto 9,* Baltimore, MD, USA.

Sacks-Davis R., McDonell K.J. and Ooi B.C. (1987). GEOQL - A Query Language for Geographic Information Systems, Internal Report 87/2, Dept of Computer Science, Royal Melbourne Institute of Technology.

Strijland P. (1990). *Icons for use on screens Part 1: Specifications.* ISO/IEC JTC 1/SC18/WG9 Working Document (4 July 1990).

Talmy L. (1990). How Language Structures Space. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space,* Las Navas, Spain

Tauber M.J. (1990) ETAG: Extended task action grammar. A language for the description of the user's task language. *Proceedings Human-Computer Interaction - INTERACT '90,* Cambridge, UK.

Westwood K. (1989). Toward the Successful Integration of Relational and Quadtree Structures in a Geographic Information System. *Proceedings National Conference - GIS - Challenge for the 1990's.,* Ottawa, Canada

Williams J.R. (1989). *Menu Design Guidelines.* ISO TC159/SC4/WG5 WD 9241-14 (17 March 1989)

Wilson P.M. (1990) Get your Desktop Metaphor off my Drafting Table: User Interface Design for Spatial Data Handling. *Proceedings 4th International Symposium on Spatial Data Handling,* Zurich, Switzerland.

Wozny L.A. (1989) The Application of Metaphor, Analogy and Conceptual Models in Computer Systems. *Interacting with Computers: The Interdisciplinary Journal of Human-Computer Interaction.* vol.1 no. 3 pp 273-283

Algebraic Optimization of Combined Overlay Operations*

Claus Dorenbeck†
Max J. Egenhofer‡
National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469, U.S.A.
CLAUS@MECAN1.bitnet
MAX@MECAN1.bitnet

Abstract

The operations necessary to combine map layers are formalized with algebraic specifications. This shows that arithmetic operations upon discrete spatial subdivisions are reduced to a single, parametric overlay operation, the actual behavior of which is determined by a value operation which combines the non-spatial attributes of the individual cells of the corresponding layers. The novel approach is the application of these formalisms to find more efficient strategies for processing several overlay operations at an implementation-independent level. Two particular strategies are investigated: (1) the elimination of equivalent subexpressions to reduce the complexity of the overlay operation and (2) the integration of several overlay operations into a single one.

1 Introduction

Spatial data models (Peuquet 1984, White 1984, Frank and Kuhn 1986, Herring 1987, Egenhofer *et al.* 1989) and spatial data structures (Peucker and Chrisman 1975, Corbett 1979, Nagy and Wagle 1979, Samet 1989) have been extensively studied in the past. More recently, the interest in the relations between spatial data models and spatial data structures has increased (Egenhofer and Herring 1991, Frank 1991b, Frank and Mark 1991, Goodchild 1991). Initial results of these investigations are:

- A *spatial data model* is the formalization of *spatial concepts* so that they can be represented in a computer.

*This work was partially funded by a grants from NSF under grant number SES 88-10917 and Intergraph Corporation, principal investigator Andrew U. Frank.

†On a leave of absence from the University of Bremen, F.R. Germany.

‡Additional support from Digital Equipment Corporation under grant number TP-765536 is gratefully acknowledged.

- A *spatial data structure* is the implementation of a particular spatial data model.
- A spatial data model may have multiple implementations with various effects on the performance and the storage requirements.
- A spatial data structure must fulfill the properties of the operations specified for the data model, in order to be a valid implementation of a spatial data model.

For example, the data model of a regular subdivision of space into squares of equal size, frequently called a *raster model*, may be implemented with different spatial data structures, such as a 2-dimensional array, run-length encoded, as a quadtree data structure, etc. (Samet 1989).

Within this framework of spatial data models and spatial data structures, a question of particular interest is, "How to describe formally the behavior of the operations?" This question covers the properties of the operations, i.e., the linkage between a data model and its various implementations, but, more importantly, also the specifications of the properties of their combinations. Note that it does not treat the actual implementation, i.e., a particular data structure or an algorithm.

Formalizations of spatial data models and GIS operations are necessary to (1) verify that an implementation, i.e., a spatial data structure, does what was set forth by the spatial data model, and (2) compare the semantics of different spatial data models (Frank 1987, Smith and Frank 1990). Non-spatial data models have been formalized, for example, by the relational algebra which specifies the behavior of the operations upon relational tables (Codd 1970, Ullman 1982), but only subsets of spatial algebras exist, e.g., for topological relationships (Egenhofer and Herring 1990) or directions (Peuquet and Ci-Xiang 1987, Chang *et al.* 1989, Frank 1991a). Each of these approaches is limited to a very specific class of operations and no attempts have been made to integrate them into a larger system.

The Map Analysis Package provided the first comprehensive collection of analytical and spatial operations on the basis of regular tessellations (Tomlin 1983, Tomlin 1990). It describes map overlay operations informally, without applying the mathematical rigor necessary to analyze the behavior of the operations, leaving ample space for ambiguous interpretations. One implementation of this *MAP algebra* describes formally these operations in the C^{++} programming language (Chan and White 1987), but lacks the definitions of the corresponding observe operations so that no axioms about the *behavior* of the operations can be formulated.

More formal approaches are based on the Euclidean plane and the representation of spatial data in terms of points, lines, and areas. A formalisation of the non set-theoretic part of Euclidean geometry (Tarski 1959) gives a collection of thirteen elementary axioms. An algebraic specification of graphic data types formally defines the semantics of a simple graphics programming language without geometric operations (Mallgren 1982). An algebra for geometric constructions, based upon well-known algebraic structures such as rings and fields (Goguen 1989), demonstrates the use of algebraic specifications for spatial objects, however, it is limited to a few, very basic constructs in plane geometry. The geo-relational algebra enhances the relational model with computational geometry operations (Güting 1988). A formalized interpretation of operations upon maps uses set operations and a construct similar to constructs in functional programming languages, such as *mapcar* in LISP, applying a user-defined function to each element of a set (Scholl and Voisard 1989).

This paper addresses the formalization of operations on regular tessellations to assess optimization strategies, particularly for combinations of overlays. The operations necessary to combine map layers are formalized with algebraic specifications and the optimization attempts to identify a more efficient combination of the initial operations. It employs algebraic methods to substitute complex combinations by simpler ones (Ullman 1982), a technique commonly employed in electrical engineering applications (Preparata and Yeh 1973) and compiler design (Aho *et al.* 1985).

The remainder of this paper is structured as follows: the next section briefly introduces the two formalisms used, i.e., algebraic specifications and decision tables. Section 3 formalizes a specific map overlay operation and then generalizes this formalism so that it becomes applicable for arbitrary overlay operations. The properties of value operations to combine layers, cell-by-cell, are analyzed in Section 4, and strategies are proposed for efficient combinations of multiple overlay operations. Section 5 summarizes the results and concludes with directions for further research.

2 Formal Methods

2.1 Algebras

Multi-sorted algebraic specifications are a tool commonly used in software engineering to describe completely and formally the behavior of complex systems (Liskov 1986). They are based on heterogeneous algebras (Birkhoff and Lipson 1970) and their extensions to multi-sorted algebras (Guttag 1977). Data algebras (Zilles 1979), using equations to define the independent properties of data structures, and abstract data types (ADTs) (Goguen *et al.* 1978) influenced today's understanding of algebraic specifications. An *algebraic specification* consists of three parts (Liskov 1986, Ehrich *et al.* 1989): (1) a set of *sorts*,[1] (2) a set of *operations* defined upon the sorts, and (3) a set of *axioms* or *equations* that specifies the behavior of the operations. Two kinds of operations are distinguished: (1) operations to create or modify an ADT, called *creators*, and (2) operations to observe some of the properties of an ADT, called *observers*. A specification is sufficiently correct and sufficiently complete in terms of its creators and observers (Guttag and Horning 1978).

The following example specifies an ADT *point* in the Euclidean plane. The ADTs *integer* and *boolean* are assumed to exist with their usual semantics. The syntax of this specification method resembles the syntax of specification-like programming languages such as Eiffel (Meyer 1988) and MOOSE (Egenhofer and Frank 1988).

[1]The term *sort* does not imply an order (sorting) over the instances. Programming languages use the less ambiguous term *type* in lieu of *sort*, however, *types* consider also the structure of the *sorts* (Cardelli and Wegner 1985) which is part of an implementation.

```
SORTS[2]       point USES integer, boolean
OPERATIONS[3]  make:  integer × integer → point
               x:  point → integer
               y:  point → integer
               isEqual:  point × point → boolean
VARIABLES[4]   i1, i2:  integer; p1, p2:  point
EQUATIONS[5]   x (make (i1, i2)) == i1
               y (make (i1, i2)) == i2
               isEqual (p1, p2) == integer.isEqual (x (p1), x (p2)) and
                                   integer.isEqual (y (p1), y (p2))
```

Specification 1: Point.

2.2 Decision Tables

A *decision table* is another method to specify formally the behavior of operations, particularly those which can be described by a series of rules. It consists of two parts: (1) a set of *conditions* which have to be satisfied simultaneously and (2) the corresponding *actions* to be taken upon the conditions (Metzner and Barnes 1977).

Decision tables are most naturally presented in the form of a table with the set of conditions being put into the upper half of the table and the set of corresponding actions underneath. Boolean values, T and F, are assigned to the conditions indicating whether or not the corresponding action should be taken. If an action is taken independent of a condition then a dash in the corresponding decision indicates *don't care.* Since the entries in conditions and corresponding actions are logically connected with *AND*, they are commutative.

Decision tables are a well-suited tool to express some spatial analysis operations which frequently use complex algebraic expressions to describe their operations and mappings. The following example demonstrates the use of a decision table to formalize a particular value operation, the *localRating*, frequently used in the MAP algebra (Tomlin 1990). LocalRating assigns to each n-tuple of values a new value. For example, the following localRating combines a layer of *altitudes* with a *vegetation* layer into a new layer *windExposure.*

- If the altitude is greater than or equal to 290 and vegetation type 0 then the wind exposure is 1.
- If the altitude is greater than or equal to 290 and vegetation types 1–3 then the wind exposure is 2.
- If the altitude is less than 290 and vegetation type 0, 1, or 3 then the wind exposure is 3.

[2]The SORTS definition includes the data type to be specified and the types it USES to describe its properties.

[3]OPERATIONS are defined by their name, the Cartesian product of the input sorts, and the sort of the result.

[4]VARIABLES describe the instances of the sorts used in the equations.

[5]The behavior of each operation is expressed by EQUATIONS in terms of equivalent observe and create operations.

- If the altitude is less than 290 and vegetation type 2 then the wind exposure is 4.

Table 1 shows a decision table which models these rules.

altitude vegetation	≥ 290 0	≥ 290 $1 \vee 2 \vee 3$	< 290 $0 \vee 1 \vee 3$	< 290 2
windExposure	1	2	3	4

Table 1: The decision table for the local rating of altitudes and vegetation [Tomlin 1989].

3 Formalizing Overlay Operations

The raster model is a particular subclass of the regular tessellations with a discrete representation of space (Egenhofer and Herring 1991, Frank 1991b). It partitions the area of interest into equally-shaped *cells* so that (1) the set of all cells forms a complete partition, called a *layer*, and (2) any pair of cells does not overlap. This section will demonstrate the use of algebraic specifications to specify formally combinations of layers.

3.1 An Overlay Example

The most common queries upon layers are based on the *map overlay* methodology, i.e., the combination of several layers into a new one (Steinitz *et al.* 1976). A simple, but specific example is to show the use of algebraic specifications for describing a particular overlay operation. The operation to be specified is the combination of the two layers with regular rectangular cells, both over the same spatial extent, in the same scale, and with the same orientation. Each cell is made from a *location* and a *value*. In this particular example, each value is an integer, with the operations equal and maximum and their usual semantics, and each location is a rectangle described by its lower-left and upper-right points (Specification 1), a creator (make), and three observe operations (Specification 2).

```
SORTS       location USES point, boolean
OPERATIONS  make:  point × point → location
            lowerLeft:  location → point
            upperRight:  location → point
            isEqual:  location × location → boolean
VARIABLES   p1, p2:  point, l1, l2:  location
EQUATIONS   lowerLeft (make (p1, p2)) == p1
            upperRight (make (p1, p2)) == p2
            isEqual (l1, l2) == point.isEqual (lowerLeft (l1),
                                               lowerLeft (l2)) and
                                point.isEqual (upperRight (l1),
                                               upperRight (l2))
```

Specification 2: Location as the Cartesian product of two points.

Cells have operations to *make* a new one and to access its components, i.e., *getLocation* and *getValue* (Specification 3).

```
SORTS       cell USES location, value
OPERATIONS  make:  location × value → cell
            getLocation:  cell → location
            getValue:  cell → value
VARIABLES   l:  location; v:  value
EQUATIONS   getValue (make (l, v)) == v
            getLocation (make (l, v)) == l
```

Specification 3: Cells.

The resulting layer contains the greater of the two values at the corresponding spatial locations (Specification 4).

```
SORTS       layer USES cell
OPERATIONS  make:  cell × cell × ... × cell → layer
            overlayMaximum:  layer × layer → layer
VARIABLES   l1, l2:  layer; c1, c2, c3:  cell
EQUATIONS   FOR EACH [c1, c2, c3] IN [l1, l2, overlayMaximum (l1, l2)]:
                location.isEqual (cell.getLocation (c3),
                                  cell.getLocation (c1)) and
                location.isEqual (cell.getLocation (c3),
                                  cell.getLocation (c2)) and
                value.isEqual (cell.getValue (c3),
                value.maximum (cell.getValue (c1), cell.getValue (c2))).
```

Specification 4: Combining two layers by selecting the maximum value.

The syntax of the equations uses a FOR EACH[5] loop (Liskov *et al.* 1981) to apply an operation to all elements of a set (Backus 1978), i.e., all cells which are part of, or IN, a layer. Actually, this is an observe operation upon a layer returning the cells in the aggregate one after another. Simultaneous loops over multiple aggregates group the parts and the aggregates pairwise between brackets so that the n-th part in on bracket corresponds with the n-th aggregate in the other.

This set of specifications for layers, cells, rectangles, and integers completely formalizes the behavior of this particular overlay operation.

- Layers are combined by applying a particular operation to corresponding cells, i.e., cells with the same spatial location.
- The same value operation is applied to all cells of a layer.
- The value combination of cells preserves the locations of the cells, i.e., the location of each cell in the resulting layer is the same as the one of the cells combined.

3.2 A Generalized Overlay Operation

The previous specification can be generalized so that it holds for other overlay operations as well. Such a generic specification is based upon the definition of a generalized value type, a superclass of all possible sorts which may characterize the non-spatial properties of a cell.

A value type must provide operations to compare two values for equality (`isEqual`) and to `combine` values (Specification 5). The specification of its `create` operation is `DEFERRED` (Meyer 1988), because it depends upon the particular value type used.

```
SORTS      value USES boolean
OPERATIONS create:  DEFERRED → value
           isEqual:  value × value → boolean
           combine:  value × value ×...× value → value
```

Specification 5: A generic value.

Likewise, the location specification may vary for different shapes of cells. Besides the `make` operation, the ADT location must provide an operation to compare two locations for equivalence (`isEqual`) (Specification 6).

```
SORTS      location USES boolean
OPERATIONS make:  DEFERRED → location
           isEqual:  location × location → boolean
```

Specification 6: A generic location.

[5]Not to be confused with the for-all quantifier, $\forall$, commonly used in calculus.

The specification of the ADT cell as the Cartesian product of location and value stays unchanged (Specification 3). The modified ADT layer has a single overlay operation with varying implementations depending on the value operation used to combine corresponding cells. The FOR EACH loop runs over the sets of all cells in all layers, indicated by c_i and l_i, respectively (Specification 7).

```
SORTS       layer USES cell
OPERATIONS  make:  cell × cell ×...× cell → layer
            overlay:  layer × layer ×...× layer × value.combine → layer
VARIABLES   c_i, c_n:  cell; l_i:  layer
EQUATIONS   FOR EACH [c_i, c_n] IN [l_i, overlay (l_i, value.combine)]:
                 location.isEqual (cell.getLocation (c_n),
                                   cell.getLocation (c_i)) and
                 value.isEqual (cell.getValue (c_n),
                                value.combine (cell.getValue (c_i)))
```

Specification 7: A parametric layer.

The behavior of any overlay operation is expressed by a particular operation upon the values of individual cells (`value.combine`). The usage of a variable argument over value operations reduces the specification to a single, generic operation. `Combine` is similar to the operators *apply* (Scholl and Voisard 1989) and λ (Güting 1988) in other formalizations.

The generalized overlay specification reveals that the characteristics of these overlay operations are exclusively determined by the operation combining several values. Conversely, the properties of the value operation immediately map onto the properties of the overlay operation. For arithmetic overlay operations, it is sufficient to consider each layer as a set of cells, i.e., no topological relationships among the cells are used. Since the values are combined over the same location, the overlay operation—in terms of relational algebra (extended with arithmetic capabilities) (Ullman 1982)—is (1) an equijoin over the same location (Frank 1987) followed by (2) an arithmetic operation `combining` the values of corresponding location and (3) a projection of the locations and the combined value.

4 Optimization

An overlay operation over multiple layers results in a new layer which, in turn, may be used as an argument in another overlay operation. Frequently, many overlay operations are combined this way to perform a more complex operation (Tomlin 1990). While sophisticated spatial data structures may efficiently implement an individual overlay operation, they generally provide only little support for improving the processing of a series of overlays. It will quickly become time consuming to process sequentially each overlay operation by producing an intermediate layer after each operation. In lieu of immediately performing each operation, it is more efficient to evaluate first the entire operation and identify an execution strategy which predicts the shortest processing time. Similar considerations within the relational algebra to gain better performance for complex, combined operations led to the area of *query optimization* (Ullman 1982). To date, only few attempts have been made to improve systematically spatial query processing (Hudson 1989, Ooi and Sacks-Davis 1989). Current overlay processors calculate interactively one overlay at a time (Pazner *et al.* 1989), though there have been recently attempts to pursue more efficient processing strategies (Yost and Skelton 1990). To improve the overlay operations of several layers, two strategies are investigated: (1) to identify equivalent sub-expressions so that they can be computed only once, and (2) to integrate several individual overlay operations into a single one. Both strategies will be investigated subsequently.

4.1 Notation

The uppercase Greek letter *omega* (Ω) will be used to denote overlay. Its arguments are (1) the ordered set of layers $layer_1, \dots, layer_n$ with $n > 0$, and (2) a particular *combination* operation (Equation 1).

$$\Omega_{combination}(layer_1, \dots, layer_n) \tag{1}$$

The combination operation may be a function, such as *max* or *average*, or a decision table. For instance, the value combination specified in decision table 1 is applied to the layers altitudes and vegetation, resulting in the layer windExposure (Equation 2).

$$windExposure := \Omega_{Table\ 1}\ (altitude, vegetation) \tag{2}$$

4.2 Equivalent Overlay Operations

A first step during processing the combination of overlays is to identify those sequences of operations that occur several times so that they need to be executed only once. The goal for such an overlay optimizer is to find equivalent, but more efficient expressions, i.e., expressions which yield the same result within less time. This strategy requires a formal knowledge of equivalent expressions. Mathematics has the notion of properties of combinations of operations to describe whether two expressions are equivalent or not. Most familiar are the commutative, associative, and distributive laws, e.g., for the combinations of sets with the operations union and intersection. Likewise, the combination of layers with various overlay operations may be described by their commutative (Equation 3), associative (Equation 4), and distributive (Equation 5) properties.

$$\Omega_{combination}(layer_1, layer_2) = \Omega_{combination}(layer_2, layer_1) \tag{3}$$

$$\begin{aligned}&\Omega_{combination}(\Omega_{combination}(layer_1, layer_2), layer_3) = \\ &\quad \Omega_{combination}(layer_1, \Omega_{combination}(layer_2, layer_3))\end{aligned} \tag{4}$$

$$\begin{aligned}&\Omega_{combination_1}(layer_1, \Omega_{combination_2}(layer_2, layer_3)) = \\ &\quad \Omega_{combination_2}(\Omega_{combination_1}(layer_1, layer_2), \Omega_{combination_1}(layer_1, layer_3))\end{aligned} \tag{5}$$

The specification of the generalized overlay operation (Specification 7) demonstrated that an overlay varies only over different value operations; therefore, the properties of the combinations of overlay operations can be based upon the properties of the corresponding value operation. For instance, the combination of three layers is associative if and only if the value operation is associative as well:

$$\begin{aligned}&\Omega_{combination}(\Omega_{combination}(layer_1, layer_2), layer_3) = \\ &\quad \Omega_{combination}(layer_1, \Omega_{combination}(layer_2, layer_3)) \qquad (6)\\ \Leftrightarrow \\ &value.combine(value.combine(v_1, v_2), v_3) = \\ &\quad value.combine(v_1, value.combine(v_2, v_3)) \qquad (7)\end{aligned}$$

Since the overlay operations depend completely upon the corresponding value operations, they can be optimized by only considering the value operations in the same sequence as the corresponding overlay operations. Equivalent overlay operations can be found by analyzing the properties of the value operations. These properties are described in the axioms of the specifications of the values. For example, given a complex query containing the following expressions:

$$\ldots\Omega_{add}(layer_1, \Omega_{add}(layer_2, layer_3))\ldots\Omega_{add}(layer_3, \Omega_{add}(layer_2, layer_1))\ldots \qquad (8)$$

The axioms of the particular value specifications may provide the necessary information about the properties of the add operation, e.g.,

```
SORTS       value
OPERATIONS  add:  value × value → value
VARIABLES   v₁, v₂, v₃:  value
EQUATIONS   add (v₁, v₂) == add (v₂, v₁)
            add (v₁, (add (v₂, v₃)) == add (add (v₁, v₂), v₃))
```

Specification 8: Commutative and associative properties of the value operation *add.*

Based upon these axioms it can be formally analyzed whether or not these two expressions are the same. First, the overlay operation is substituted by the corresponding operations upon values (Equations 9 and 10).

$$\Omega_{add}(layer_1, \Omega_{add}(layer_2, layer_3)) \Rightarrow value.add(v_1, value.add(v_2, v_3)) \qquad (9)$$
$$\Omega_{add}(layer_3, \Omega_{add}(layer_2, layer_1)) \Rightarrow value.add(v_3, value.add(v_2, v_1)) \qquad (10)$$

Then the axioms are applied. With the associative law, Equation (10) is transformed.

$$value.add(v_3, value.add(v_2, v_1)) = value.add(value.add(v_3, v_2), v_1) \qquad (11)$$

Finally, the commutative law is applied twice.

$$\begin{aligned}value.add(value.add(v_3, v_2), v_1) &= value.add(v_1, value.add(v_3, v_2))\\ &= value.add(v_1, value.add(v_2, v_3)) \qquad (12)\end{aligned}$$

Equation (12), the equivalent for (10), is the same as (9), i.e., the two subexpressions in Equation (8) are the same and, therefore, only one of them must be executed.

4.3 Integration of Multiple Overlay Combinations

A second strategy to reduce the execution time of a complex combination of overlays is to integrate several overlay operations into a single, equivalent one, i.e.,

$$\Omega_{combination_1}(layer_1, \Omega_{combination_2}(layer_2, layer_3)) \Rightarrow \Omega_{combination_3}(layer_1, layer_2, layer_3) \quad (13)$$

Again, the specification of the generalized overlay operation (Specification 7) was fundamental in tackling this problem. It shows that this integration means to move a value operation, `value.operation₂`, from the inner FOR EACH loop into the outer loop and combine `value.operation₁` with `value.operation₂` into `value.operation₃`. The validity of such combinations can be checked with the axioms specifying the value ADTs.

```
FOR EACH ($t_1$, $t_2$) IN (A,
        FOR EACH ($t_3$, $t_4$) IN (B, C)
                DO value.operation$_2$)
        DO value.operation$_1$
$\Rightarrow$
FOR EACH ($t_1$, $t_2$, $t_3$) IN (A, B, C)
        DO value.operation$_3$
```

An alternative approach to this symbolic optimization is the use of decision tables to evaluate the combinations. Given the sets of values on each layer, the decision tables can be applied to analyze the property of the combination of operations.

The following example demonstrates such an integration. Four layers, A_1, A_2, B_1, and B_2, with the four respective sets of values, {2, 4, 8}, {6, 10}, {3, 4}, and {1, 2, 3}, should be combined such that

$$result := \Omega_{min}(\Omega_{average}(A_1, A_2), \Omega_{Table\ 2b}\ (B_1, B_2)) \quad (14)$$

The decision table 2 shows the combinations for the two inner overlays.

A_1	2	2	4	4	8	8
A_2	6	10	6	10	6	10
X_1	4	6	5	7	7	9

B_1	3	4	4
B_2	-	$1 \vee 3$	2
X_2	2	1	3

Table 2: (a) $X_1 := average(A_1, A_2)$ and (b) $X_2 := (B_1, B_2)$.

Table 3 shows the result of the combination of the two intermediate results with the operation *min*.

X_1	4	4	4	6	6	6	5	5	5	7	7	7	9	9	9
X_2	2	1	3	2	1	3	2	1	3	2	1	3	2	1	3
X	2	1	3	2	1	3	2	1	3	2	1	3	2	1	3

Table 3: $X := min(X_1, X_2)$.

The sequence of operations may be expressed by a single table. Its condition part contains the Cartesian product of the values in the four layers and the action part has the corresponding values of the combinations (Table 4).

A_1	2	2	2	4	4	4	2	2	2	4	4	4	8	8	8	8	8	8
A_2	6	6	6	6	6	6	10	10	10	10	10	10	6	6	6	10	10	10
B_1	4	3	4	4	3	4	4	3	4	4	3	4	4	3	4	4	3	4
B_2	$1 \vee 3$	–	2	$1 \vee 3$	–	2	$1 \vee 3$	–	2	$1 \vee 3$	–	2	$1 \vee 3$	–	2	$1 \vee 3$	–	2
X	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3

Table 4: $X := min(average(A_1, A_2), Table\ 2b(B_1, B_2))$.

Table 4 can be simplified by combining columns with the same actions, e.g.,

A_1	$2 \vee 4 \vee 8$	$2 \vee 4 \vee 8$	$2 \vee 4 \vee 8$	$2 \vee 4 \vee 8$	$2 \vee 4 \vee 8$	$2 \vee 4 \vee 8$
A_2	6	10	6	10	6	10
B_1	4	4	3	3	4	4
B_2	$1 \vee 3$	$1 \vee 3$	–	–	2	2
X	1	1	2	2	3	3

Table 5: $X := min(average(A_1, A_2), Table\ 2b(B_1, B_2))$.

Further integrations (over the values of A_2) and the substitutions of disjunctions which cover the entire domain by the value *don't care* reduce the value operation to an operation which is independent of the two layers A_1 and A_2 (Table 6); therefore, the entire overlay operation may be reduced to the combination of the two layers B_1 and B_2.

A_1	–	–	–
A_2	–	–	–
B_1	4	3	4
B_2	$1 \vee 3$	–	2
X	1	2	3

Table 6: The simplified decision table for $\Omega_{min}(\Omega_{average}(A_1, A_2), \Omega_{Table\ 2b}(B_1, B_2))$.

The decision table also indicates in which order the two layers should be processed. The value of a layer needs not be examined if the result is independent of it. For example, it is more efficient to execute $\Omega_{Table\ 2b}(B_1, B_2)$ than $\Omega_{Table\ 2b}(B_2, B_1)$. In the first case, the result of half of the operations is determined by just examining B_1, because the outcome of the combination with value 3 is independent of the value at the corresponding location in B_2. If the value is 4 then the corresponding value in B_2 must be examined as well. On the other hand, if the converse operation is executed then always the values of both layers must be processed.

5 Conclusion

Rigid formal methods have shown to be effective tools to identify optimization strategies for combinations of overlay operations. The algebraic specification of a generalized overlay operation for tessellations revealed that

- a layer may be considered a set of cells, each consisting of a location and a value, and
- arithmetic overlay operations over layers can be broken down into a value operation to be performed for each cell of a layer or tuple of corresponding cells in several layers, similar to the application of a function to a whole set in functional programming.

Since overlay operations are founded upon value operations, it is possible to map the considerations about best execution plans for operations onto considerations about the combination of value operations. Two particular ways of optimizing several overlay operations have been investigated:

1. the use of axiomatic description of the value operations to identify whether or not two combinations of value operations are equivalent. Faster executions of combinations of overlays are possible, because such equivalent subexpressions can be substituted by the result of one single overlay operation.
2. the use of decision tables, representing the characteristics of value operations, to integrate several overlay operations. This method can be applied if the sets of values of all layers are known. The integration reduces any combination of overlay operations into a single one and is most effective if the number of conditions is small. Decision tables are less suitable for large sets of conditions, because the tables grow multiplicatively before reduction.

The results obtained demonstrated the usefulness of the approach. Further investigations are necessary to build sophisticated query optimizers for raster GIS's. The present work, intentionally, excluded geometric operations on cells, e.g., those which exploit the neighborhood relationship between cells. Within the formal framework provided it is now possible to study their behavior to formalize geometric operations on tessellations.

6 Acknowledgements

Thanks to Andrew Frank for his useful comments on earlier versions of this paper and to Kelly Chan, Werner Kuhn, Hans-Peter Kriegel, and Alan Saalfeld for stimulating discussions.

References

A. Aho, R. Sethi, and J. Ullman. Compilers: Principles, Techniques, and Tools. Addison-Wesley Publishing Company, Reading, MA, 1985.

J. Backus. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. Communications of the ACM, 21(8):613–641, August 1978.

G. Birkhoff and J. Lipson. Heterogeneous Algebras. Journal of Combinatorial Theory, 8:115–133, 1970.

L. Cardelli and P. Wegner. On Understanding Type, Data Abstraction, and Polymorphism. ACM Computing Surveys, 17(4):471–552, April 1985.

K. Chan and D. White. Map Algebra: An Object-Oriented Implementation. In: International Geographic Information Systems (IGIS) Symposium: The Research Agenda, Vol. II, pages 127–150, Arlington, VA, November 1987.

S.K. Chang, E. Jungert, and Y. Li. The Design of Pictorial Databases Based Upon the Theory of Symbolic Projections. In: A. Buchmann, O. Günther, T. Smith, and Y. Wang, editors, Symposium on the Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science, Vol. 409, pages 303–323, Springer-Verlag, New York, NY, July 1989.

E.F. Codd. A Relational Model for Large Shared Data Banks. Communications of the ACM, 13(6):377–387, June 1970.

J. Corbett. Topological Principles of Cartography. Technical Report 48, Bureau of the Census, Department of Commerce, 1979.

M. Egenhofer and A. Frank. MOOSE: Combining Software Engineering and Database Managemenst Systems. In: Second International Workshop on Computer-Aided Software Engineering, Advance Papers, Cambridge, MA, May 1988.

M. Egenhofer, A. Frank, and J. Jackson. A Topological Data Model for Spatial Databases. In: A. Buchmann, O. Günther, T. Smith, and Y. Wang, editors, Symposium on the Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science, Vol. 409, pages 271–286, Springer-Verlag, New York, NY, July 1989.

M. Egenhofer and J. Herring. A Mathematical Framework for the Definition of Topological Relationships. In: K. Brassel and H. Kishimoto, editors, Fourth International Symposium on Spatial Data Handling, pages 814–819, Zurich, Switzerland, July 1990.

M. Egenhofer and J. Herring. High-Level Spatial Data Structure. in: D. Maguire, D. Rhind, and M. Goodchild, editors, Geographical Information Systems: Overview, Principles, and Applications, Longman Scientific and Technical, London, 1991 (in press).

H.-D. Ehrich, M. Gogolla, and U. Lipeck. Algebraic Specifications of Abstract Data Types (in German). B.G. Teubner, Stuttgart, 1989.

A. Frank and W. Kuhn. Cell Graph: A Provable Correct Method for the Storage of Geometry. In: D. Marble, editor, Second International Symposium on Spatial Data Handling, pages 411–436, Seattle, WA, 1986.

A. Frank. Overlay Processing in Spatial Informaion Systems. In: N. Chrisman, editor, AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography, pages 16–31, Baltimore, MD, March 1987.

A. Frank. Qualitative Spatial Reasoning about Cardinal Directions. In: D. Mark and D. White, editors, Autocarto 10, Baltimore, MD, March 1991a.

A. Frank. Spatial Concepts, Geometric Data Models and Data Structures. Computers and Geo-Sciences, 1991b (in press).

A. Frank and D. Mark. Language Issues for Geographical Information Systems. in: D. Maguire, D. Rhind, and M. Goodchild, editors, Geographical Information Systems: Overview, Principles, and Applications, Longman Scientific and Technical, London, 1991 (in press).

J. Goguen, J. Thatcher, and E. Wagner. An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types. In: R. Yeh, editor, Current Trends in Programming Methodology, Prentice-Hall, Englewood Cliffs, NJ, 1978.

J. Goguen. Modular Algebraic Specification of Some Basic Geometrical Constructions. In: D. Kapur and J. Mundy, editors, Geometric Reasoning, pages 123–153, MIT Press, Cambridge, MA, 1989.

M. Goodchild. A Geographical Perspective on Spatial Data Models. Computers and Geo-Sciences, 1991 (in press).

R. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In: J. Schmidt, S. Ceri, and M. Missikoff, editors, Advances in Database Technology—EDBT '88, International Conference on Extending Database Technology, Venice, Italy, Lecture Notes in Computer Science, Vol. 303, pages 506–527, Springer Verlag, New York, NY, 1988.

J. Guttag. Abstract Data Types And The Development Of Data Structures. Communications of the ACM, 20(6):396–404, June 1977.

J. Guttag and J. Horning. The Algebraic Specification of Abstract Data Types. Acta Informatica, 10:27–52, 1978.

J. Herring. TIGRIS: Topologically Integrated Geographic Information Systems. In: N. Chrisman, editor, AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography, pages 282–291, Baltimore, MD, March 1987.

D. Hudson. A Unifying Database Formalism. In: ASPRS/ACSM Annual Convention, pages 146–153, Baltimore, MD, April 1989.

B. Liskov, R. Atkinson, T. Bloom, E. Moss, J.C. Schaffert, R. Scheifler, A. Snyder. CLU Reference Manual. Lecture Notes in Computer Science, Vol. 114, Springer-Verlag, New York, NY, 1981.

B. Liskov and J. Guttag. Abstraction and Specification in Program Development. MIT Press, Cambridge, MA, 1986.

W. Mallgren. Formal Specification of Graphic Data Types. ACM Transactions of Programming Languages and Systems, 4(4):687–710, October 1982.

J. Metzner and B. Barnes. Decision Table Languages and Systems. Academic Press, New York, NY, 1977.

B. Meyer. Object-Oriented Software Construction. Prentice Hall, New York, NY, 1988.

G. Nagy and S. Wagle. Geographic Data Processing. ACM Computing Surveys, 11(2):139–181, June 1979.

B. Ooi and R. Sacks-Davis. Query Optimization in an Extended DBMS. In: W. Litwin and H.-J. Schek, editors, Third International Conference on Foundations of Data Organization and Algorithms (FODO), Lecture Notes in Computer Science, Vol. 367, pages 48–63, Springer-Verlag, New York, NY, June 1989.

M. Pazner, K.C. Kirby, and N. Thies. MAP II: Map Processor—A Geographic Information System for the Macintosh. John Wiley & Sons, New York, NY, 1989.

T. Peucker and N. Chrisman. Cartographic Data Structures. The American Cartographer, 2(2):55–69, 1975.

D. Peuquet. A Conceptual Framework and Comparison of Spatial Data Models. Cartographica, 21(4):66–113, 1984.

D.J. Peuquet and Z. Ci-Xiang. An Algorithm to Determine the Directional Relationship Between Arbitrarily-Shaped Polygons in the Plane. Pattern Recognition, 20(1):65–74, 1987.

F. Preparata and R. Yeh. Introduction to Discrete Structures. Addison-Wesley Publishing Company, Reading, MA, 1973.

H. Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Company, Reading, MA, 1989.

M. Scholl and A. Voisard. Thematic Map Modeling. In: A. Buchmann, O. Günther, T. Smith, and Y. Wang, editors, Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, CA, Lecture Notes in Computer Science, Vol. 409, pages 167–190, Springer-Verlag, New York, NY, July 1989.

T. Smith and A. Frank, Report on Workshop on Very Large Spatial Databases. Journal of Visual Languages and Computing, 1(3):291–309, 1990.

C. Steinitz, P. Parker, and L. Jorden. Hand-Drawn Overlays: Their History and Prospective Uses. Landscape Architecture, 66(8):444–455, 1976.

A. Tarski. What is Elementary Geometry? in: L. Henkin, P. Suppes, and A. Tarski, editors, Symposium on the Axiomatic Method, pages 16–29. North Holland, Amsterdam, 1959.

C.D. Tomlin. Digital Cartographic Modeling Techniques in Environmental Planning. PhD thesis, Yale University, 1983.

C.D. Tomlin. Geographic Information Systems and Cartographic Modeling. Prentice-Hall, Englewood Cliffs, NJ, 1990.

J. Ullman. Principles of Database Systems. Computer Science Press, Rockville, MD, 1982.

M. White. Technical Requirements and Standards for a Multipurpose Geographic Data System. The American Cartographer, 11(1):15–26, March 1984.

M. Yost and B. Skelton. Programming Language Technology for Raster GIS Modeling. In: GIS/LIS 90, pages 319–327, Anaheim, CA, November 1990.

S. Zilles. An Introduction to Data Algebras. In: D. Bjørner, editor, Abstract Software Specifications, pages 248–272, Spring-Verlag, New York, 1979.

SPATIAL OVERLAY WITH INEXACT NUMERICAL DATA

David Pullar
Surveying Engineering Department
National Center for Geographical Information and Analysis
University of Maine,
Orono, Maine, U.S.A.

Abstract

A methodology for the operation of spatial overlay is presented in this paper. A general framework for spatial overlay based on concepts in epsilon geometry is developed to cope with the problems of computational errors and handling inaccurate numerical data. These problems normally cause topological inconsistencies and generate spurious effects in the result. A mapping is defined to accommodate the edges and vertices in all spatial layers so they are unambiguously aligned within a prescribed tolerance. Geometrical arguments are given to show the correctness of this approach.

1. Introduction

Spatial overlay is an analytical tool used to integrate multiple thematic layers (coverages) into a single composite layer. Each layer is organized into a polygon-network structure where polygons are assigned to nominal, or ordinal, categories. The data may be efficiently stored using a topological data structure [Peucker and Chrisman 1975]. Spatial overlay has proven to be a very powerful tool to analyse the association among different spatial patterns and land characteristics. Despite its popularity there is practically no theory to guide the development of algorithms.

Development of programs for spatial overlay have been hindered by a number of problems related to map accuracy and computational geometry. If the input coverages are overlaid exactly, then sliver polygons are produced along different versions of the same boundary or spatially correlated boundaries represented in different layers. Slivers are an undesirable byproduct of overlay as they are meaningless and degrade further analysis and interpretation of the data [Goodchild 1978]. Other problems that arise are as follows;

i) repeated application of tolerance intersections cause objects to move outside their tolerance [White 1978],

ii) numerical instability causes topological inconsistencies [Franklin 1984],

iii) spurious affects from fuzzy creep and subtolerent segments [Guevara 1985].

One of the most significant achievements in polygon overlay software can be found in the ODYSSEY system for geographic information processing [Dougenik 1979][Chrisman 1983]. The overlay program for ODYSSEY addressed many of the problems listed above, and computed *tolerant intersections* as a solution to the sliver polygon problem [White 1978]. The primitive operation for intersection was broadened to include a tolerance parameter, and clusters of intersection points were analysed to form consistent nodes [Chrisman 1983]. The value for the tolerance parameter is related to the accuracy and scale of the input coverages. However, a problem arises when trying to overlay many coverages which have multiple accuracies. A central tenet of this paper is that the concepts and approaches used to analyze spatial data can profit from further improvement of the overlay algorithm. A general framework for map overlay which will integrate many coverages, with multiple tolerances, in a single operation is described. The overlay algorithm is similar to Zhang and Tulip [1990] in avoiding slivers by snapping less accurate points to more accurate points and not moving any point outside its tolerance. We extend this work by presenting a verifiable methodology for the overlay operation.

The proposed approach, called *map accommodation*, is based upon a simple concept of accommodating the geometry between each layer to bring them into alignment in the composite layer. Map accommodation detects and reports all types of intersections and proximities between spatial data, and then objectively analyses the data to resolve conflicts.

An outline for the paper is as follows. The next section describes issues related to geometrical intersection. Section 3 describes an algorithm for reliable polygon set operations. It is a robust algorithm, but it does not place an overall bound on positional errors introduced in the process. Section 4 gives a brief outline of the solution we propose, it aims to both bound and minimize any positional errors. Sections 5 and 6 give a more formal treatment of the overlay problem and the correctness of the proposed solution. Section 7 describes a clustering algorithm which is central to our approach. Section 8 analyses the performance of the proposed algorithm and suggests some enhancements.

2. Geometric Operations

Geometric operations on objects representing physical phenomena pose special problems for the design of computer algorithms. Hoffmann [1989] says that "practical implementations of geometric modeling operations remain error-prone, and a goal of implementing correct, efficient, and robust systems for carrying them out has not yet been attained".

Geometric operations encounter two types of errors; i) numerical errors, and ii) representational errors.

2.1 Numerical Errors

Geometric operations on polyhedral objects represented by floating-point numbers will introduce numerical errors in the result [Hoffmann 1989]. The most common numerical error is round-off error. Geometric operations use intermediate results from numerical computations to derive symbolic information. For instance, when a computed variable is less than, equal to, or greater than zero indicates if a point lies below, on, or above a line. If the difference between the values compared is less than a certain threshold ε, computations can lead to misleading results. Two broad approaches are proposed for treating numerical errors; a) compute an exact result by performing intermediate computations with a higher precision [Ottmann et al 1987], or b) determine error intervals around geometric objects and perturb a version of the input data so objects are unambiguously related to one another within their respective intervals [Hoffmann et al 1988].

2.2 Representational Errors

The coordinate descriptions for geometric objects, whether explicitly stated or not, are expressions of measurement and include some positional uncertainty. Geometric operations need to, a) capture the notion of "approximate tests", and b) to provide estimates on the accuracy of objects. Guibas and others [1989] describe a general framework for coping with errors in geometric operations called *epsilon geometry*, and show how epsilon-predicates are incorporated in geometric tests.

In a similar vein Milenkovic [1989] describes a technique, called *data normalization,* to perform reliable geometrical set operations on polyhedral objects. Data normalization acts as a preprocessing stage to resolve any topological ambiguities before performing set operations. A version of the input data is perturbed slightly to get better agreement between vertices and edges in the output overlay. The decisions on what to perturb and by how much can become very complex for arrangements of line segments.

Epsilon geometry bears some resemblance to techniques used in computer-assisted cartography. Blakemore [1984] suggested an epsilon band could represent the positional uncertainty of a digitized line, and illustrated its use to answer a point-in-polygon query. It has also been used operationally in procedures for map overlay and feature generalization. The overlay program in ODYSSEY was the first to include an epsilon tolerance to control moving the location of boundaries for the removal of sliver polygons [Dougenik 1979]. In feature generalization an epsilon tolerance is used for line filtering [Chrisman 1983] [Perkal 1966].

The approach used in epsilon geometry can cope with inaccuracies significantly larger than those introduced by numerical errors. Therefore, epsilon geometry provides a good general framework to deal with both numerical and representational errors. This chapter builds on these and other works to explore the use of epsilon geometry in polygon overlay. The next section describes the method used to compute the intersection of polyhedral objects which applies the concepts of data normalization. It is a robust algorithm, but has undesirable drawbacks we wish to improve upon.

3. Data Normalization

This section describes a published algorithm that includes some discussion of the reasoning steps involved in geometric operations. Milenkovic [1989] describes a simple method to give a definite and correct answer to geometric operations on polyhedral objects. The idea is to perturb the positions of objects, that are within a certain threshold ε, so they coincide exactly. The method is called *data normalization,* and it assures all data objects satisfy two numerical tests:

1) no two vertices are closer than a tolerance ε, and

2) no vertex is closer to an edge than a tolerance ε.

Two primitive operations are applied to the data to satisfy these conditions, *vertex shifting* and *edge cracking*. See figures 1 and 2. Vertex shifting will move one vertex to another if they are closer than tolerance ε. If a vertex is within tolerance ε to an edge, edge cracking will move the vertex to a new cracked point along the edge.

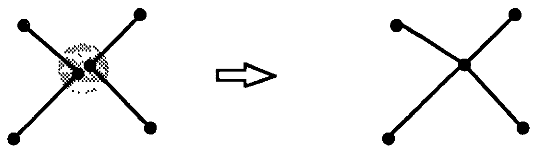


Figure 1: Vertex Shifting

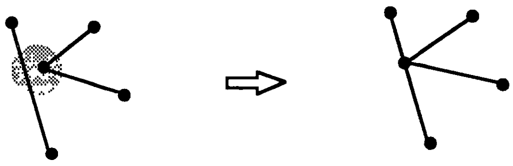


Figure 2: Edge Cracking

The algorithm described by Milenkovic initially makes two passes through the data. The first pass tests for near coincidence between vertices from the input polygons. When two vertices are found within the threshold tolerance then one vertex is shifted and identified with the other vertex. The second pass tests for the proximity of vertices to edges for the input polygons, and does edge cracking where necessary. Edge cracking may introduce further near coincidences. Hence, the algorithm is reiterated until both the numerical tests for data normalization are satisfied. With each iteration slight perturbations accumulate and may lead to positional alterations larger than ε, this is called *creep*. For example, in the left diagram of Figure 3 the vertices u_1 and v_1 are within tolerance ε of edge (u_0v_0). The right diagram shows the results after edge cracking, now the vertices u_2 and v_2 are within ε of the new edge (u_1v_1) which in turn calls for further cracking. Edge cracking may continue in a cascading fashion so that points along edge (u_0v_0) will migrate outside their given tolerance.

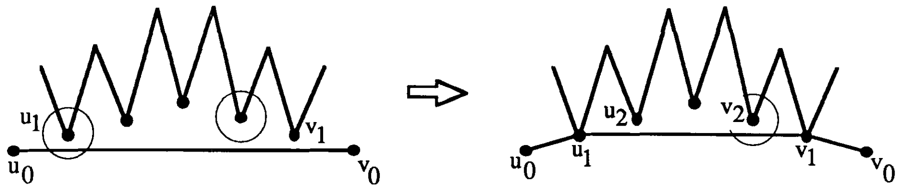


Figure 3: Creep introduced by edge cracking.

This algorithm will compute output polygons with a valid planar topology, so in this sense it is robust. However, positional error will accumulate with each iteration so the procedure cannot place a constant bound on the extent polygons are perturbed. The next section describes another approach which avoids creep.

4. Proposed Solution

This paper has used the ideas behind data normalization and adapted them to a new approach to overlay called *map accommodation.* This section gives a brief description of the mechanics of our approach to map overlay, and shows how a clustering procedure is incorporated with the primitive operations for vertex shifting and edge cracking. Latter sections will give a more formal and detailed description.

4.1 Issues

The two drawbacks to the algorithm described in the previous section are;

i) the perturbations are performed in an arbitrary fashion, and

ii) it does not prevent creep.

First, there is no objective evaluation of relations between the geometric primitives (vertices and edges) to guide what gets snapped to what. It is a greedy algorithm which accommodates primitives of one polygon to another as violations to normalization conditions occur. We propose an algorithm which objectively evaluates the proximities between geometric primitives to guide the accommodation process. Like the overlay program in ODYSSEY [Dougenik 1979], a clustering strategy is used to minimize the perturbations to the data. The benefit of this approach is it promotes a stable map topology.

The stability of map topology and its relation to a discrete surface model is investigated by Saalfeld [1987]. Measures of stability, or robustness in the topological structure, are defined in terms of a geometrical measure of the closeness of primitives. We propose a strategy to cluster primitives based upon proximity, such that primitives are clumped together to minimize perturbations and provide the greatest separability between cluster centers. In this way we expect greater stability in the overlay transformation.

Second, primitives can migrate from their original locations by a significant distance for certain degenerate configurations. This was discussed briefly in the previous section and, Milenkovic demonstrates a case where a valid polygon can collapse to a point. To avoid the effects of creep we perform clustering in a special way to bound the perturbations.

4.2 Brief Outline of Algorithm

The remainder of this section gives a simple description of map accommodation, and latter sections will go into greater detail. The accommodation algorithm accepts as input N layers of data structured in an topological format. To start, we check all vertex-vertex proximities. If any two vertices are within a geometrical tolerance to one another they are reported in a list. Cluster analysis is performed on the elements in this list to determine consistent output vertices. By consistent we mean vertices satisfy the normalization condition, that is; i) no two vertices are closer than the tolerance, and ii) no vertex is moved greater than its tolerance. In effect, we have performed the task of vertex shifting to accommodate the vertices among the layers.

Next, we check all vertex-edge proximities. If a vertex is found to be within the given geometrical tolerance to another edge, then the vertex and its closest point along the edge are reported in a list. Cluster analysis is again performed on this list to determine consistent output vertices. Any internal points to edges that are clustered to other points are treated as a cracked edge.

Finally, we check all edge-edge intersections. The composite layer may now be assembled in a straight-forward manner.

4.3 Example

The process just described is illustrated by overlaying two simple geometric objects shown in figure 4.

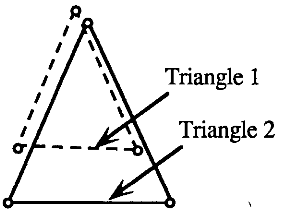


Figure 4: Two figures to be overlaid.

In figure 5, the highlighted vertices are found to be within the geometrical tolerance to one another, and are clustered and identified with a single vertex. In figure 6, the highlighted areas show edges found to be within the tolerance to a vertex. The edges are broken at the closest point to the offending vertex, this point and the vertex are clustered and identified with a single vertex. Now the geometry for the two triangles are accommodated to common vertices and edges. It is a now a relatively simple problem to compute their Boolean intersection. From the resulting figure we can answer approximate tests concerning the coincidence or inclusion of vertices and edges.

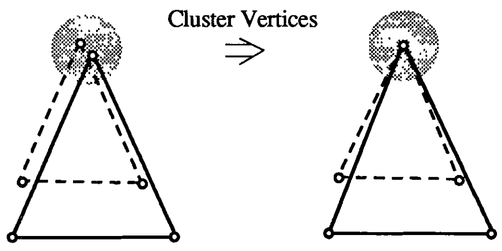


Figure 5: Vertex Shifting

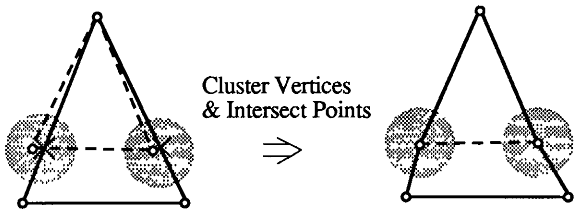


Figure 6: Edge cracking

This section has presented a very simple description of map accommodation. Subsequent sections will give a more formal and detailed treatment of the process.

5 Accommodation Conditions

This chapter is mainly concerned with how the overlay transformation affects the geometry of objects. Map accommodation is required to satisfy five geometric conditions which validate the result of polygon overlay. These conditions form the basis for the design of the map accommodation algorithm.

The basic geometric primitives are vertices and edges, and an associated tolerance parameter called epsilon. A map layer is denoted by the triple $\{V,E,\in\}$, where;

V is the set of vertices v representing points in the plane,

E is the set of edges e made up of ordered pairs of vertices, and

$\in$ is the epsilon parameter ε associated with the vertices or edges.

To establish a convention, we will use lower case symbols to denote primitives of a set and upper case to denote the sets. For example, e_i^R is the i-th element in the R-th set of edges, or in other words $e_i^R \in E^R$. An element e_i^R has an associated tolerance denoted ε_i^R.

A vital part of the overlay process is to accommodate the geometric primitives from all input layers to resolve topological ambiguities. A number of input layers will be mapped to a single composite layer. If a primitive is not within the epsilon tolerance to another primitive then it remain unchanged. However, if it is necessary to accommodate primitives from different layers then this may cause the combining of primitives or the insertion of new primitives. The five conditions for map accommodation determine what sort of changes are allowed. These conditions examine spatial proximities and are defined in terms of the Euclidean distances *d* between primitives. Formally, we define the *accommodation mapping* for 1..N input layers to one composite layer as,

$$\{V^1,E^1,\in^1\}, \{V^2,E^2,\in^2\}, \ldots \{V^N,E^N,\in^N\} \Longrightarrow \{V',E',\in'\}$$

such that the following accommodation conditions are satisfied;

A1) v_i^R is moved to v_j' iff $d(v_i^R, v_j') < \varepsilon_i^R$,

A2) for any two v_i', v_j' implies $d(v_i', v_j') > \text{minimum}(\varepsilon_i', \varepsilon_j')$,

A3) a cracked point p on e_i^R is moved to v_j' iff $d(p, v_j') < \varepsilon_i'$,

A4) for any v_i' and point p on any e_j' implies $d(p, v_i') > \text{minimum}(\varepsilon_i', \varepsilon_j')$,

A5) no two e' intersect except at a common vertex v'.

An infinite number of mappings will satisfy these condition. Therefore, another constraint is imposed to minimize any positional alterations. This is achieved by using cluster analysis to objectively chose the output vertices to which other vertices are moved. The next section describes the way cluster analysis is used in map accommodation and presents informal arguments to show how the above conditions are satisfied.

6 Accommodation Process

Central to the accommodation process is the clustering procedure. Clustering will analyze points and replace the points which are agglomerated with their weighted centroid. The weight for a point is related to its associated epsilon parameter. For now, we only define the properties of the clustering method and leave the description of the clustering algorithm for the next section. The input to cluster analysis is a set $\{P, \in\}$, where;

P is a set of points p in the plane, and

$\in$ is the epsilon parameter ε associated with each point.

Formally, clustering is a mapping between sets as,

$$\{P, \in\} \Longrightarrow \{P', \in'\},$$

where p_j' is the weighted centroid for agglomerated points $p_i \in P$, and ε_j' is selected as the minimum of the $\varepsilon_i \in \in$, and such that the following clustering properties hold;

C1) for any p_i clustered to p_j' implies $d(p_i, p_j') < \varepsilon_i$, and

C2) for any two p_i', p_j' implies $d(p_i', p_j') > \text{minimum}(\varepsilon_i', \varepsilon_j')$.

Based on the given properties of a clustering, we describe each stage in the accommodation technique and give informal arguments to show they satisfy the accommodation conditions. This is followed by an outline of each step in the algorithm. The three major stages of the accommodation technique are;

1. Vertex shifting,
2. Edge cracking,
3. Edge intersection.

The order of execution for each stage is designed to detect and resolve correlation between boundaries. Correlated boundaries will demonstrate a similar pattern of curvature. This correlation will be most prevalent at corresponding terminal points and break points along boundary lines. Therefore, detection of coincident vertices should proceed first. Figure 7(a) shows two correlated lines with areas of vertex coincidence highlighted. Figure 7(b) shows the result after vertex shifting. Subsequent detection of vertex to edge proximities will detect correlation along the boundary line. Figure 7(b) also highlights areas of vertex to edge coincidence, and figure 7(c) shows the result after edge cracking. The final stage will detect clear cases for two edges crossing.

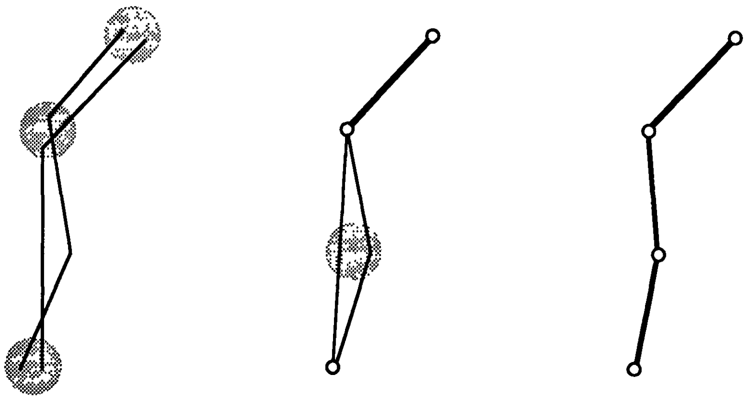

(a) Detect vertex coincidence (b) After vertex shifting (c) After edge cracking

Figure 7: Map accommodation for two lines

6.1 Vertex Shifting

The task of vertex shifting is to detect vertices that are approximately coincident, and then compute consistent output vertices. One pass is made through the input data performing pairwise comparisons between vertices in each layer. When the distance between two vertices is discovered to be less than the sum of their tolerances they are reported in a list. This list serves as input to the clustering algorithm. Clusters are computed and any affected vertices are identified with a new vertex at the respective cluster centroid.

Clustering will satisfy properties (C1) and (C2), which is sufficient to prove that conditions (A1) and (A2) for the accommodation mapping (given in §5) are satisfied. That is, no vertex is perturbed outside it epsilon tolerance, and vertices are separated by at least the minimum epsilon tolerance.

6.2 Edge Cracking

Edge cracking detects vertices that lie approximately on an edge, and then computes consistent output vertices which are inserted along the appropriate edge. One pass through the data is required to find all vertices near edges. A vertex v_i is considered near to an edge e_j, by first finding point p as the orthogonal projection of v_i onto e_j, when $d(v_i,p) < (\varepsilon_i+\varepsilon_j)$. Any affected vertices and cracked edges are reported in the list.

There does exist degenerate cases that require additional checking. In certain geometric configurations cracking an edge will cause further edge cracking. Figure 8 shows such a case, v causes e_1 to be cracked at p_1 which in turn causes e_2 to be cracked at p_2. The later point is called an *induced intersect point,* these constructs were first identified in the overlay part of the ODYSSEY program [Harvard 1983].

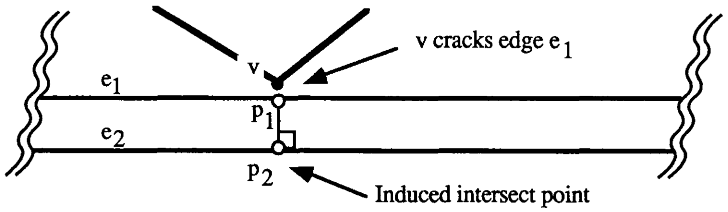


Figure 8: Edge e_1 is cracked at p_1 which may cause edge e_2 to be cracked at p_2

To guard against these degenerate cases requires an additional test. All new vertices located along a cracked edge are tested against all other edges. If any induced intersect points are discovered they are reported in the list. This list serves as input to the clustering algorithm. Clusters are computed and any affected vertices are identified with a new vertex at the respective cluster centroid.

Clustering properties (C1) and (C2) are sufficient to guarantee conditions (A1) and (A3) for the accommodation mapping (given in §5) are satisfied. Edge cracking will not violate condition (A2) because the only way for two vertices to move close to each other is if they are cracked by an edge between them, and therefore they must already have been discovered and evaluated in the edge cracking procedure. By searching for and including induced intersect points in the cluster analysis will guarantee there are no further violations to condition (A4). Therefore, conditions (A1) to (A4) for the accommodation mapping are satisfied.

6.3 Edge Intersection

Edge intersection identifies a common vertex at the point where two or more edges cross. One pass through the data is required to find all cases where edges cross at internal points. Note all intersections are computed without ambiguities since vertices and edges now satisfy conditions (A1) to (A4). However, by creating a new vertex at the intersection point may cause a violation to condition (A4). Figure 9 illustrates the degenerate case that needs to be treated. An additional test for further edge cracking is required (with the new intersection points only) to detect induced intersections. Again, all intersection points and induced intersect points serve as input to cluster analysis.

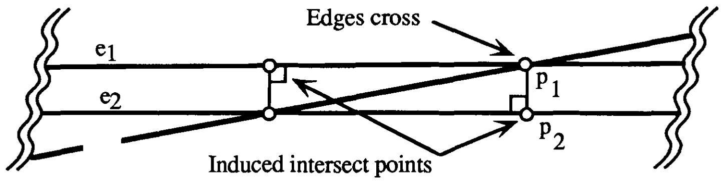


Figure 9: Edges e_1 and e_2 intersect at p_1, which may cause edge e_3 to be cracked at p_2

When two edges cross they are cracked at a common point. This point is identified as a common vertex for both edges, so condition (A5) of the accommodation mapping is now satisfied. After this, all the conditions for the accommodation mapping are satisfied and we can proceed to rebuild the topological structure for the composite layer in a reasonably straight forward manner by tracing polygonal paths.

6.4 Accommodation algorithm

An outline of the algorithm for accommodation is presented. Accommodation calls the cluster analysis procedure. For simplicity it is assumed clustering will satisfy conditions (C1) and (C2) for any input, and then details for the clustering algorithm are explained in the next section.

PRELIMINARY

The algorithm accepts as input any number of layers, each composed of a set of edge-paths. A tolerance parameter is associated with each edge, therefore tolerances may also vary within layers. For convenience we shall denote the R-th layer in our set terminology as $\{V^R,E^R,\epsilon^R\}$. The algorithm needs a data structure to store intersect points to be clustered, this is called MSET.

ALGORITHM - overlay of N layers of chains

Map-accommodation: $\{V^1,E^1,\epsilon^1\}, \{V^2,E^2,\epsilon^2\}..\{V^N,E^N,\epsilon^N\} \Longrightarrow \{V',E',\epsilon'\}$

Step 1. Do pairwise comparison of vertices v_i^R, v_j^S where R≠S, and report all pairs within the tolerance, i.e. $d(v_i^R, v_j^S) < (\varepsilon_i^R+\varepsilon_j^S)$, to the set MSET.

Step 2. Perform a clustering on the points in MSET. Identify all agglomerated vertices v with the appropriate cluster centroid v' and assign the minimum ε to ε'.

Step 3. Do a pairwise comparison of vertices and edges v_i^R, e_j^S where R≠S, and report pairs within the tolerance, i.e. let p be point on e_j^S perpendicular to v_i then $d(v_i^R,p) < (\varepsilon_i^R+\varepsilon_j^S)$, to the set MSET.

Step 4. Repeat Step 3 for the newly cracked points, i.e. the cracked points p on e_j^S, to find any induced intersect points, and report these pairs to the set MSET.

Step 5. Perform a clustering on the points in MSET. Identify all agglomerated vertices v and cracked points on e with the appropriate cluster centroid v' and assign the minimum ε to ε'.

Step 6. Do pairwise intersection of edges e_i^R, e_j^S where R≠S, and report pairs that intersect (at an interior point on both edges) to the set MSET.

Step 7. Repeat Step 3 for intersect points, i.e. the point p on e_i^R and e_j^S, to find any induced intersect points, and report these pairs to the set MSET.

Step 8. Perform a clustering on the points in MSET. Identify all intersect points on e with the appropriate cluster centroid v′ and assign the minimum ε to ε'.

7. Clustering

A central part of our approach to the accommodation process is the clustering algorithm. The previous section defined the properties of a clustering, this section describes how clustering is performed. A clustering problem is defined as a partition of a finite set into n disjoint sets based upon minimizing an objective function. The objective function is typically some proximity measure to bring out intrinsic structure in the data. The complexity of obtaining a global optimal solution is NP-complete for n-partitions (n>2) in two or more dimensions [Brucker 1978]. This is not computationaly feasible, so a suboptimal heuristic solution is proposed.

The task at hand is to describe; i) the objective function used to measure proximity between clusters, and ii) the strategy used to form partitions of the data.

7.1 Proximity Matrix

The proximity matrix represents an index of similarity (or dissimilarity) measures between pairs of clusters. The most commonly used criteria for computing these similarity measures is a *square-error criteria,* and is based on minimizing the square error between component points and their computed cluster centroid [Jain & Dubes 1988]. This is similar to minimizing the within-cluster variation and maximizing the between-cluster variation. We show how this criteria is adapted to clustering points with a geometrical tolerance.

The weighted arithmetic mean for a set of values x_i and their associated weights w_i is;

$$\bar{a}_w = \frac{\sum_{i=1}^{N} a_i w_i}{\sum_{i=1}^{N} w_i} .$$

The weighted mean vector for a cluster K, denoted m^k, is defined as the *cluster centroid.* This is computed by the weighted arithmetic mean for ordinates of the cluster points. If a coordinate is denoted by the pair {x,y} then the weighted coordinate centroid m^k is denoted as $\{\bar{x}_w, \bar{y}_w\}$. The weight is related to the positional uncertainty associated with a point, and is defined as the inverse of the square of the epsilon tolerance, i.e. $w = 1/\varepsilon^2$.

The square-error for the kth cluster with n_k members is given as;

$$e_k^2 = \sum_{i=1}^{n_k} \left(x_i^k - m^k\right)^T \left(x_i^k - m^k\right) = \Sigma_{i=1,n_k} d(x_i^k, m^k)^2 .$$

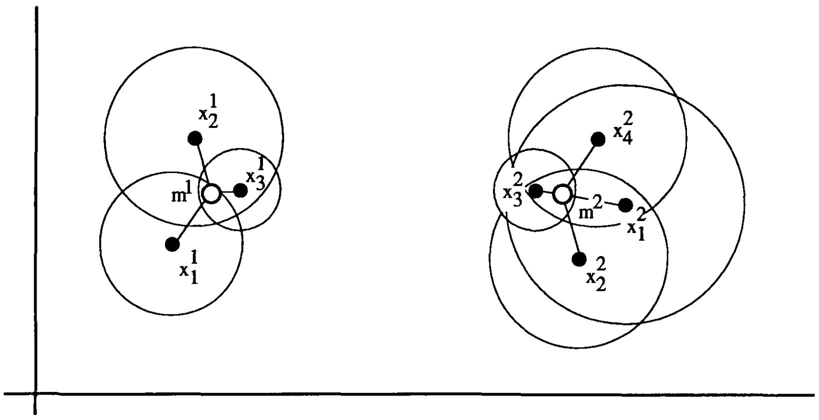


Figure 10: Distances used in computing square-error

A minimum variance partition is defined as a clustering which minimizes the sum of the square-error for a fixed number of N clusters, that is by minimizing the expression;

$$E_N^2 = \Sigma_{k=1,N}\ e_k^2 .$$

The similarity measure computed in this chapter must additionally obey the following constraints;

1) the distance between a cluster centroid and all its member points is not greater than the given geometrical tolerance, and
2) the function should seek to minimize the number of clusters.

To fulfill the first requirement, we define the *bounded-square-error* for the kth cluster C_k as;

$$\bar{e}_k^2 = e_k^2 \quad \text{if } \forall_{x_i \in C_k}\ d(x_i, m^k) \le \varepsilon_i$$
$$= \infty \quad \text{otherwise,}$$

where ε_i is the geometrical tolerance associated with x_i. This says that we are only interested in points within the given geometrical tolerance to the centroid, otherwise the square-error can be some very large number.

In the second requirement, we need to limit the final number of clusters by gradually merging clusters. The idea is to find the minimum number of clusters which satisfy conditions (C1) and (C2). At the same time the partitioning chosen should yield the minimum value for E_N^2 using a bounded-square-error criteria. A solution to this problem is computationally not feasible. In fact, it requires examining the power set of all points. The next section describes a suboptimal solution to the problem.

7.2 Clustering Strategy

We have already shown that an optimal partitioning based on minimizing an objective function is not computationally feasible. Therefore, a selection strategy is used to reduce the number of partitions evaluated to achieve a "reasonable" approximation. The selection process is designed to converge to a local minima of the objective function. Jain and Dubes [1988] give an extensive discussion on the factors involved in cluster strategies. The major choices are between hierarchical and partitional schemes. Hierarchical clustering schemes organize the data into a nested sequence of groups. Partitional clustering schemes successively determine partitions of clusters such that points are moved between clusters in an effort to improve a criteria function. The major disadvantages we see for a partitional clustering are; i) it is very sensitive to a hill-climbing solution, ii) it is designed to solve for a fixed number of partitions. A hierarchical procedure is not as sensitive to a hill-climbing solution; but as Jain and Dubes state, its most desirable feature is in modeling the global structure of the data.

An agglomerative algorithm for hierarchical clustering is proposed. It starts by placing each point into an individual cluster. A proximity matrix made up of similarity measures between clusters is computed. This matrix is interpreted to merge two or more clusters at each level in the hierarchy. The process is repeated to form a sequence of nested clusters. The bounded-square-error criteria will terminate when all values in the proximity matrix are infinity. This provides a reasonable solution to minimizing the number of clusters.

The algorithm needs an appropriate data structure to store clusters and their respective member points. Operations for merging clusters and for finding which cluster a particular point is in must be supported. A very efficient data structure called a MERGE-FIND ADT is described in Aho, Hopcroft and Ullman [1985] for this purpose.

8. Analysis of Algorithm

We can analyse the computational cost for the accommodation mapping in each of the three steps by examining the complexity for; i) geometrical intersection, and ii) clustering.

First, geometrical intersection involves the pairwise comparison between primitives in the various layers. If there are M layers each having T primitives (interpret this as either the number of vertices or edges) then it requires $(MT)^2$ proximity comparisons between primitives in a brute force approach. The number of points reported will depend on the degree of spatial correlation between primitives in different layers. We estimate the worst case occurs when all the layers are the same giving 2MT points.

A plane sweep solution to geometrical intersection [Preparata and Shamos 1985] is unsuitable because the sweep invariant requires a strict order between edges and the vertical sweep line. A modified sweep technique using a band sweep is used in the ODYSSEY program [White 1978], and this is claimed to work well. An alternative method using a griding technique [Franklin 1989] was adopted for our implementation.

The edges were organized into edge-cell pairs by testing if the band (given by the epsilon tolerance about an edge) overlapped a grid cell. Then a brute force method was used to compute tolerance intersection within each cell. Performance tests on experimental data, which assume a uniform distribution of line segments, show favorable execution times for a grid partitioning technique compared to the plane sweep technique [Pullar 1990].

Second, the computational cost for clustering is extremely high using a brute force approach. Let N=2MT, then it would require N(N-1)/2 computations to construct the proximity matrix for a set of intersect points. To process this matrix for clustering requires $N^2(N-1)/4$ computations in the worst case, i.e. all N points merge to a single cluster. Therefore, the computational complexity of the algorithm is of order $O(N^3)$ in the worst case. Day and Edelsbrunner [1984] offer an improvement on this by efficient determination of nearest neighbors in the clustering algorithm, they describe an algorithm of $O(N^2 \log N)$ in the worst case. This was still felt to be an unacceptable cost.

In our implementation, the efficiency of clustering was improved by incorporating the grid partitioning technique in the algorithm. An alternative clustering procedure, which in principle works the same as an agglomerative algorithm, is used in collaboration with the uniform grid. Assuming points are uniformly distributed over the coverage, a nearest-neighbor search can be localized using a grid superimposd over the data. The technique is described in Murtagh [1983], and an upper-bound for the expected time complexity is reported to be O(N log N). If the grid resolution is no smaller than the maximum epsilon tolerance then a nearest-neighbor search may be localized to the current grid cell and its adjacent group of grid cells. In our experiments the partitioning techniques had a very satisfactory average behavior and exhibited an O(N) cost behavior. Further details of the algorithm will be published in a future article.

9. Conclusion

The main objective of this paper was to develop a methodology for map overlay which overcomes problems of computational errors and handling numerical data of an uncertain pedigree. A technique used to perform reliable geometrical set operation in solid modelling systems, called data normalization, is adapted to the map overlay problem. We show how the proposed technique, called map accommodation, will prevent creep in the geometry of primitives and promote stability in map topology. We have also presented informal arguments which demonstrate the correctness of this approach.

The advantage of our approach is it breaks the complex task of map overlay into simpler tasks which require simple data structures for implementation. The key operations for reporting intersections and spatial proximities between primitives, and clustering points involve a significant computational workload. Therefore, an efficient approach needs to be incorporated in the all stages of the algorithm to provide satisfactory performance. We propose the use of a grid partitioning technique.

Acknowledgements

I would like to express my appreciation to Renato Barrera for the advice and ideas he contributed towards this work. Support for this project is provided by Prime Wild GIS Inc., and from the U.S. National Science Foundation through the NCGIA.

References

Aho A., Hopcroft J. and Ullman J., 1985; *Data Structures and Algorithms*. Addison-Wesley Publishing Co., Reading, MA.

Blakemore M., 1984; *Generalization and Error in Spatial Databases*. Cartographica, Volume 21. pp.131-139

Brucker P., 1978; *On the Complexity of Clustering Problems*. In: Optimierung und Operations Research, Editors R.Hen, B.Korte and W.Olletti, Springer, Berlin.

Chrisman N., 1983; *Epsilon Filtering: A Technique for Automated Scale Changing*. Proceedings 43rd Annual Meeting of ACSM, Washington, DC. pp.322-331

Day W. and Edelsbrunner H., 1984; *Efficient Algorithms for Agglomerative Hierarchical Clustering Methods*. Journal of Classification, 1. pp.7-24

Dougenik J., 1980; *WHIRLPOOL: A program for polygon overlay*. Proceedings Auto-Carto 4, Vol.2, Reston, VA. pp.304-311

Dutton G., 1978; *Harvard Papers on Geographical Information Systems*. Addison-Wesley Publishing Co., Reading, MA.

Franklin W.R., 1984; *Cartographic Errors Symptomatic of Underlying Algebra Problems*. 1st International Symposium on Spatial Data Handling, Zurich. pp.190-208

Franklin W.R., 1989; Uniform Grids: *A Technique for Intersection Detection on Serial and Parallel Machines*. Proceedings of Auto-Carto 9, Baltimore, Maryland.

Goodchild M., 1978; *Statistical Aspects of the Polygon Overlay Problem*. In: Harvard Papers on Geographical Information Systems, Vol.6, Editor G. Dutton 1978.

Guevara J., 1985; *Intersection Problems in Polygon Overlay*. Unpublished paper, available from author through E.S.R.I., Redlands, CA.

Guibas L., Salesin D. and Stolfi J., 1989; *Epsilon Geometry: Building Robust Algorithms from Imprecise Computations*. Proceedings 5th Annual ACM Symposium on Computational Geometry, West Germany.

Harvard, 1983; *WHIRLPOOL Programmer Documentation*. Harvard Computer Graphics Laboratory, Cambridge, MA.

Hoffmann C., 1989; *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, CA.

Jain A. and Dubes R., 1988; *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ.

Milenkovec V., 1989; *Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic*. In: Geometrical Reasoning, Editors D. Kapur and J. Mundy, MIT Press, Cambridge, MA.

Murtagh F., 1983; *Expected-Time Complexity Results for Hierarchic Clustering Algorithms Which Use Cluster Centres*. Information Processing Letters, 16. pp.237-241

Ottmann T., Thiemt G. and Ullrich C., 1987; *Numerical Stability of Geometric Operations*. Proceedings 3rd ACM Symposium on Computational Geometry. Waterloo, pp.119-125

Perkal J., 1966; *An Objective Approach to Map Generalization.* Discussion Paper 10, Ann Arbor MI, Michigan Inter-University Community of Mathematical Geographers.

Peucker T. and Chrisman N., 1975; *Cartographic Data Structures.* American Cartographer, Vol.2. pp.55-69

Preparata F. and Shamos M., 1985; *Computational Geometry.* Springer-Verlag, New York.

Pullar D., 1990; *Comparative Study of Algorithms for Reporting Geometrical Intersections.* 4th International Symposium on Spatial Data Handling, Zurich.

Saalfeld A., 1987; *Stability of Map Topology and Robustness of Map Geometry.* Proceedings of Auto-Carto 8, Baltimore, Maryland.

White D., 1978; *A Design for Polygon Overlay.* In: Harvard Papers on Geographical Information Systems, Vol. 6, Editor G. Dutton 1978.

Zhang G. and Tulip J. 1990; *An Algorithm for the Avoidance of Sliver Polygons and Clusters of Points in Spatial Overlay.* 4th International Symposium on Spatial Data Handling, Zurich. pp.141-150

A Diagnostic Test for Error in Categorical Maps

Nicholas Chrisman and Marcus Lester
CHRISMAN@max.u.washington.edu & MARCUSL@max.u.washington.edu
Department of Geography DP 10, University of Washington
Seattle, Washington 98195 USA

ABSTRACT
A test based on exhaustive overlay of two categorical maps provides a description of error distinguished into the likely sources of that error (a diagnosis of the error). The results of the overlay are characterized by geometric, topological and attribute criteria to separate the most likely positional errors from the attribute errors. This paper applies the proposed test to a simple land cover map, which was replicated by a second interpreter. Results diagnose the positional inaccuracy and misidentifications common in such a GIS layer. Adopting this test will target the efforts of a producer's quality control functions, and it will also clarify fitness for the particular uses contemplated by others.

Preamble

A great quantity of geographic information, including maps of land use, soils, geology, property ownership and other phenomena, are represented in the form of categorical maps. A test for categorical maps is required to understand their fitness for use. Beyond a simple accuracy figure, a test should provide an indication (a diagnosis) of which component of the map might need correction or quality control attention.

For many years, cartographers, remote sensing experts and others have made do without tests or with tests that provide much less diagnostic information than a comprehensive test. The test developed here uses polygon overlay, not point sampling. Such a test is specifically mentioned in the US Proposed Standard for Digital Cartographic Data [Part III, 4.3.3].

> "4.3 Attribute Accuracy
> ... Accuracy tests for categorical attributes can be performed by one of the following methods. All methods shall make reference to map scale in interpreting classifications.
> ...
> 4.3.3 Tests based on Polygon Overlay
> A misclassification matrix must be reported as areas. The relationship between the two maps must be explained; as far as possible, the two sources should be independent and one should have higher accuracy." (Morrison, 1988, p. 133)

Despite this explicit reference in the standard, there is no complete specification for such a test. Furthermore, the standard does not discuss the diagnostic results which are possible.

This paper presents a new test for the accuracy of categorical maps. Rather than examining previously studied alternatives, this paper presents the case for the new test, using a worked example. In the conclusions, the paper will generalize beyond the specific case.

The Example

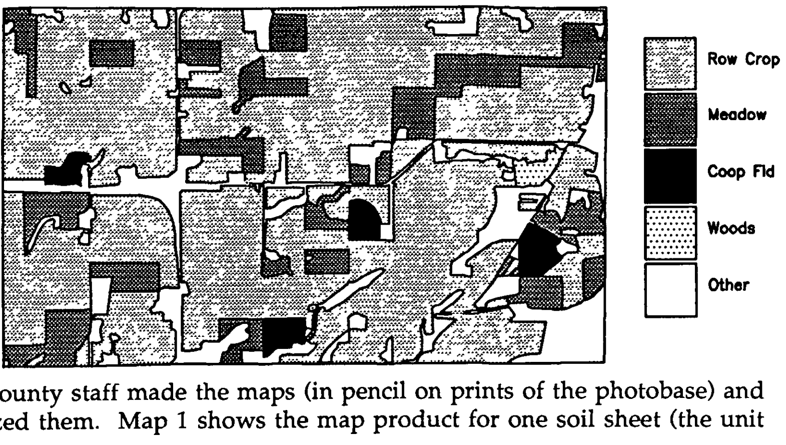
The example for this paper derives from the efforts of the Dane County, Land Records Project to generate a Soil Erosion Control Plan (Ventura, 1988). A more complete description of the project and its products has been presented in a number of publications (Chrisman and others, 1984; Niemann and others, 1987). In producing this plan, the project could rely on many layers of existing mapping, but land cover was not readily available. For the purposes of the plan, land cover requirements were relatively simple, leading to five categories:

Row Crop	Row planted crops, particularly corn and soybean
Meadow	Pasture and crops such as alfalfa and hay
Coop	Cooperator fields (in row crops or meadow)
Woods	Forested areas in rural use (not including housing)
Other	all non-rural uses plus wetlands, water, etc.

The plan needed to separate its realm of interest, rural agricultural land use, from the non-agricultural (suburban and urban). Thus, the general purpose cover category of Other included wetlands, roads, subdivisions, golf courses, industrial and commercial uses. The Woods category applies to area out of crop use – whole woodlots, not single trees. The other three categories deal with the active agricultural uses. Cooperator fields cover those areas with existing soil conservation agreements between the farmer and the conservation agencies. In any particular year, a cooperator field would be in either a row crop or meadow.

Once the conservation staff developed the categories for mapping, they had to acquire photography. The US Agricultural Stabilization and Conservation Service (ASCS) takes a color 35mm slide of each section (square mile) in Wisconsin (and many other agricultural states) each crop year to verify compliance with various federal programs. These photos are not strictly controlled photogrammetric products, but they offered color, more timely coverage and greater detail than the higher altitude products available from other sources. The project decided to use the 1982 ASCS slides to identify land cover categories and to map the results on the photographic base produced for the Dane County Soil Survey.

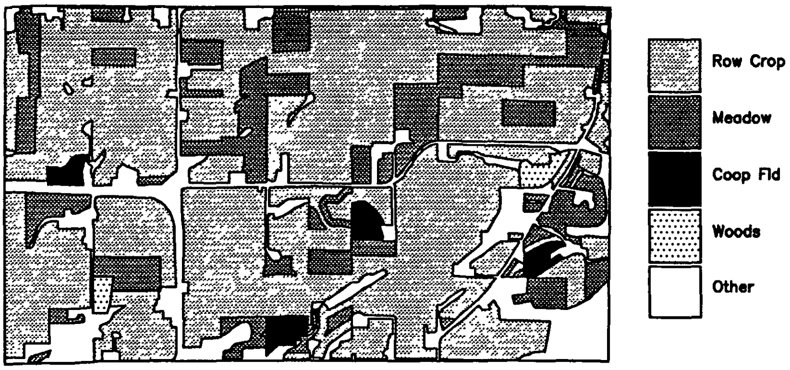
Map 1: Land Cover by Interpreter 1



The County staff made the maps (in pencil on prints of the photobase) and digitized them. Map 1 shows the map product for one soil sheet (the unit of original compilation). The University of Wisconsin–Madison team assisted in verifying the topological consistency and related operations. By summer 1983, one township (out of 35) was mapped. This pilot stage was used to demonstrate the capabilities required to complete the county plan. Much of the investigation dealt with the economics of data preparation and digitizing (Chrisman and others, 1984, p. 33-37).

So far, this process is uneventful. A local government group was making do, producing a map product to fill a project need without a large appropriation. This stage should also have included a test of accuracy in order to determine that the product was fit for the intended use. Most applications teams, being sure of their own work, forge ahead without such testing. It is the purpose of this paper to describe how a testing process could assist in the operational decisions of the GIS user. Developing such a test is, of course, a matter for theory and research. Much of the paper concentrates on the development of important ramifications of the test.

Map 2: Land Cover by Interpreter 2



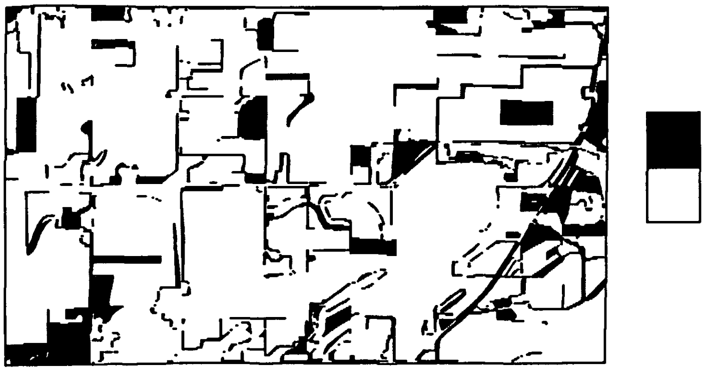
In summer 1984, another person reproduced the interpretation, according to the same rules and using the same materials (see Map 2). The second interpreter was a graduate student with some years experience as a conservationist in a nearby county. Such a test would be most authoritative based on an independent source of higher accuracy, but that would require simultaneous acquisition of another source of photography, imagery or field reports which cannot be mobilized in retrospect.

Thus, this test began with two maps of the same scale. As a test, it provides a measure of the deviation between two trials. If both interpreters agree, it shows that the classification can be reproduced reliably. When they differ, it may be due to error on either part, but the first interpreter had somewhat more field experience with this specific area.

Polygon Overlay

Map 3 shows the result of overlaying Map 2 onto Map 1. Areas where the two interpreters agreed are not shaded, while the disagreement is dark. The individual polygon boundaries have been suppressed to permit small and thin features to show in the printed format.

Map 3: Disagreement between Interpreters



The areas of the polygons created by the overlay can be crosstabulated by the categories from the two source maps in the form of a matrix called a misclassification matrix by the Proposed Standard for Digital Cartographic Data Quality (Morrison, 1988, p.133), as in Table 1.

Table 1: Misclassification Matrix (hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	1110.9	82.5	.9	.8	57.8
Meadow	17.5	212.3	.04	.2	35.8
Coop Fld	4.0	3.6	32.6	.03	2.0
Woods	.9	.3		10.2	2.1
Other	32.7	11.6	.1	7.5	212.3

From Map 3 and Table 1, it is clear that the two sources are in rough agreement. 85.8% of the area covered by the two maps falls into the diagonal of the matrix, meaning that the two interpreters agreed. For the purposes of the USGS program of land use and land cover mapping, 85% correct was set as a standard (Fitzpatrick-Lins, 1978). Thus, this example demonstrates a result at the low end of acceptability (though many remote sensing products fall much below this threshold). These overlay results were reported earlier (Chrisman, 1987) with a verbal interpretation of the reasons for the particular errors. This paper reports on an analytical procedure to decompose the error detected by overlay. The basis for such a test has been described in earlier publications (Chrisman, 1989a; 1989b); this paper reports actual results and some extensions which developed from this trial.

Outline of the Test Procedure

The basic theory behind the test distinguishes positional error from attribute (classification) error. The first arises from uncertainty in the location of a boundary, while the second arises from lack of agreement in the categories mapped. Some researchers deny the utility of this distinction. To them, a categorical map is too much of a fiction to merit the attention otherwise attached to map error for continuous surfaces. While some categorical maps may contain dubious elements, the political and administrative requirements for GIS continue to specify sharp distinctions in a fuzzy world. A test for categorical maps, such as the land cover example presented above, is sorely needed.

This section describes the procedure applied to separate the positional errors from the attribute errors. The following sections explain the rationale for these decisions, using the example as illustration.

Figure 1: Flow of test applied to each overlay polygon

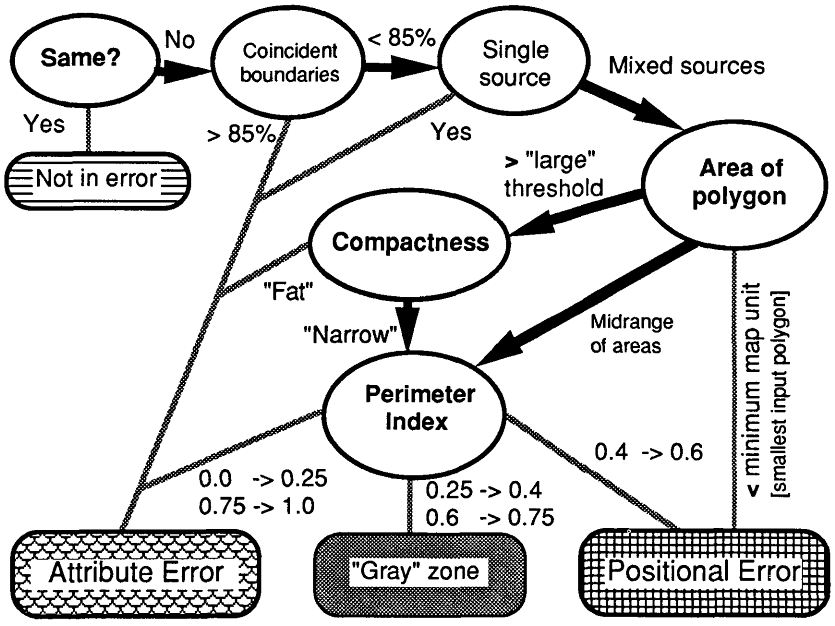


Figure 1 shows the steps involved in this test. Sequentially, each polygon is examined and fit into one of four resultant categories. There are a number of numerical parameters involved; each one may be adjusted, but the diagram shows the particular values used in this application. The first step simply decides if it represents an error. For a simple test, the attributes must be identical, in more complex cases this decision may require more information. Second, if both sources coincide to form the bulk of the polygon's boundary, then the error must be in attributes (there is not enough linework in disagreement). The parameter used here was more than 85% of the perimeter. Third, if all the non-coincident lines come from one source, it is attribute error since it must be a whole "island" of different thematic attribute, and not an area where polygons partially overlap. Fourth, polygons with area less than a "minimum map unit" are judged to be positional in nature, since they are too small to have been identified on the input layers. This threshold may be the smallest area mapped on the input layers (the procedure adopted here) or some other minimum area parameter which may be appropriate for a given test.

Fifth, for those polygons whose area is greater than a different "large" threshold (the square of a given "minimum discrimination distance" parameter), a minimum compactness value is calculated, and the compactness of the polygon is compared to this calculated minimum. This compactness value is a measure of polygon shape based upon Unwin's S_2 (1981). It is reformulated to allow a single calculation from a polygon's area and perimeter:

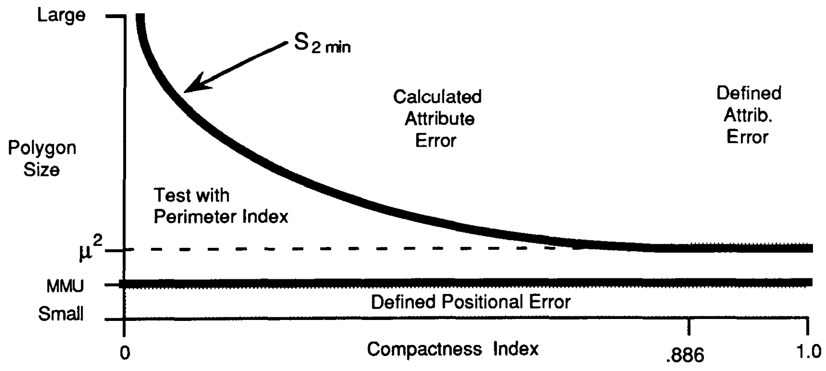
$$S_2 = 2(\pi a/p^2)^{0.5} \qquad (1$$

The minimum compactness is that of a rectangle with area equal to the polygon's area and one side equal to the minimum discrimination distance:

$$S_{2\ min} = (\pi a)^{0.5} / (\mu + (a/\mu)) \qquad (2$$

where μ = minimum discrimination distance (in this case μ is 1/8 inch on the original maps). Any polygon that is both larger than μ^2 and more compact than $S_{2\ min}$ is at once large enough and wide enough to have been identifiable on the input layers, so it is judged to be an attribute error. Essentially, this creates a sliding scale. Relatively less compact polygons can fall into the attribute error category if they are relatively large.

Figure 2: Classification by size and compactness.



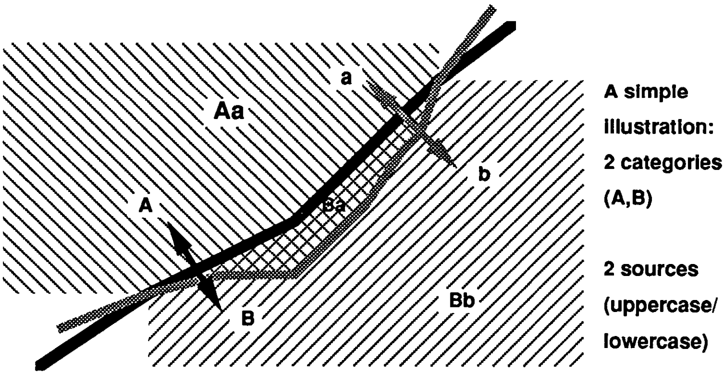
For the sixth and final stage of the test, a perimeter index is calculated for all polygons not previously classified. The perimeter index, discussed more fully below, is a ratio of the perimeter from one source to the total of the non-coincident perimeter from both sources. With this index, each remaining polygon falls into one of three categories: attribute, ambiguous (gray zone) or positional error.

Explanation of the test procedures

In applications of polygon overlay, it has long been known that "slivers" can fill up computer storage and clutter the analytical procedures (see for example, Goodchild, 1978; Cook, 1983). In this case, however, the slivers provide a clue to the origin of errors. Slivers have been identified in the past by their size and by their shape, being generally small and narrow. Narrowness is usually interpreted by human visual pattern recognition, which is difficult to quantify for complicated map features. One analytical approximation of narrowness is compactness. Compactness indices, typically ratios of perimeter to area, (see Unwin, 1981) are unreliable measures of very narrow shapes for this purpose. Because perimeter increases dramatically with line sinuosity and with inner rings of polygons, a large polygon may have a compactness index similar to a sliver.

The purpose of the test is not solely to isolate those polygons commonly called slivers. The purpose is to test the accuracy of the categorical map. Slivers are simply one form of commonly recognized error which serve as indicators of positional differences. Each of the components mentioned above; size, compactness and narrowness play a role in the proposed test, alongside a measure of "perimeter contribution". The measure of shape described in Formula 1 and 2 above serves well in distinguishing relatively compact polygons, but it is not reliable for other circumstances.

Figure 3: A typical positional error



In Figure 3, the classical sliver has some clear distinguishing characteristics. It is "small"; it is narrow; it is not compact, but these criteria are either scale-specific or could falsely identify complex features as described above. Other distinguishing characteristics might be proposed. For example, one sliver tends to engender another, in a sequence along the "true" line (Goodchild, 1978). Thus, slivers would have two possible identifying characteristics. Slivers should be topologically adjacent and the nodes at either end should be four valent (be formed by the geometric intersection of two straight lines). Both of these criteria are relatively difficult to implement for a number of reasons. First, slivers will occur at different positions, sometimes near true nodes (usually three valent for non-parcel data). It may be difficult to separate these true nodes from the sliver nodes. Second, polygon overlay is a messy business, involving the vagaries of floating point hardware (see Douglas, 1974; Dougenik, 1980). While a pure sliver might have four valent nodes at either end, the calculations of the intersection might discover a coincident section, creating two three valent nodes. Additionally, instead of random fluctuations, a long narrow sliver may be produced by a uniform (or one-sided) misinterpretation of a thematic boundary. Such a sliver may have no adjacent slivers. These same difficulties make the topological criterion difficult to manage, as well.

Another measure of slivers is required. Observing Figure 3, the linework from the two sources is nearly equal in length. In general, a sliver is an area bounded by one line from one source and a second line from the other source that are both intended to represent the same feature. In addition, other forms of positional error which do not exhibit other special characteristics of slivers also show the same balance between sources. Working backwards from the results of the test, those overlay polygons whose boundaries come from the two sources in approximately the same amounts are more likely to be positional errors. For sources of equivalent scale, the perimeter from the two sources will be very close to equality. Of course, when one source records much more detail, the perimeter may be much longer to enclose the same area. This form of cartographic texture has been related to fractal measures. In those cases, there should be more perimeter from the detailed source.

The perimeter index is a ratio which compares the perimeter from one source to the sum of the perimeters from each source (Equation 3).

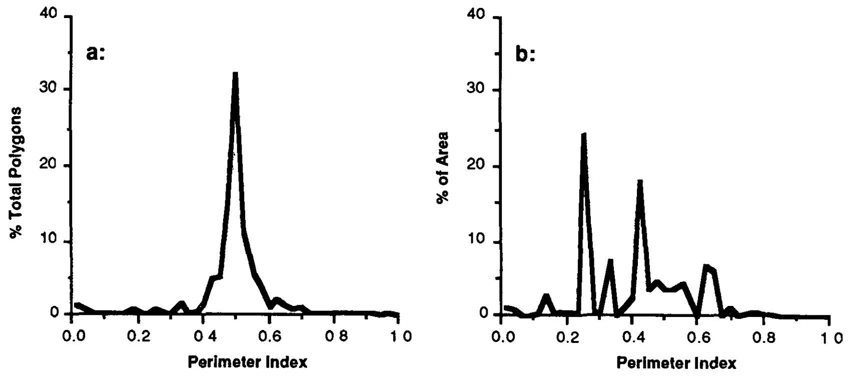
$$\text{Perimeter index} = a / (a+b) \qquad (3$$

where: a = length of chains from source A only;
b = length of chains from source B only.

In this formula, perimeter does not include those sections of a polygon's border which come from both sources (those lines which the overlay process finds to be coincident).

This index falls into the range from zero to one, with 0.5 as the result for the pure sliver (subject to the concern about scale discussed above). The index can be calculated with either map as Source A (the numerator of the ratio), which will yield values reflected around 0.5. The two versions are equivalent as long as the interpretation of the index is symmetric around 0.5. Figure 4 shows the observed distribution of the perimeter index for the land cover test presented earlier. In Figure 4 and all subsequent diagrams, the vertical axis represents percentage of the relevant total, in order to standardize the presentations.

Figure 4: Distribution of perimeter index (all polygons)
a: by percentage of count of polygons; b: by area of polygons



The pattern in Figure 4 (particularly part b) is somewhat clouded because it includes all polygons. While there is a central tendency between .4 and .6 in the number of polygons, much of the total map area comes from a few large polygons. The overlay generates 681 polygons, 106 of which are not errors. As shown in Map 3, these 106 polygons have 85.8% of the area, hence they dominate Figure 4b.

Figure 5: Distribution of index (tested polygons only)
a: by percentage of tested polygons; b: by percentage of area tested

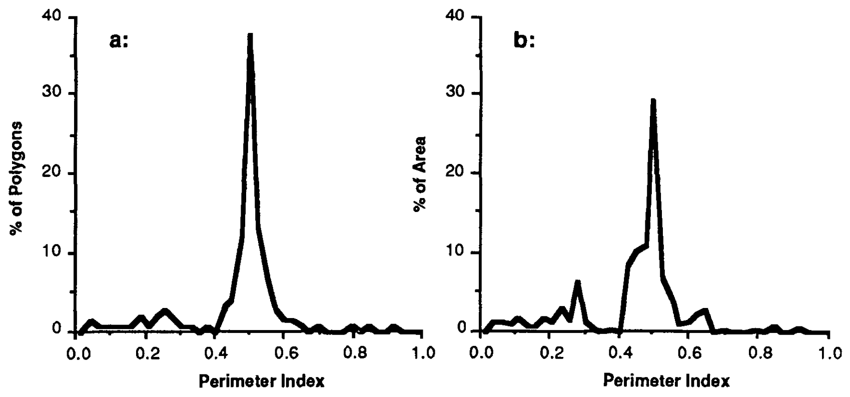
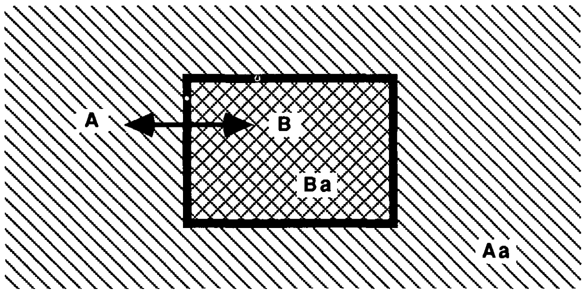


Figure 5 only tabulates the 153 polygons actually subjected to the perimeter index test, as described above. These are the polygons which are not classified by the earlier parts of the test. The largest number of overlay polygons (Figure 5a) fall near the center of the index, around 0.5. In both diagrams, the distribution is similar, and the values of .4 and .6 fall near obvious breakpoints in the distribution.

Figure 6 is a diagram of an area which is misclassified by one interpreter; the linework comes from just one source. In the diagram, the uppercase letters show a distinction on one source, but the lowercase letters show that the whole region is classified "a" in the other source. The line comes completely from one source. These errors are classified as attribute errors by the single source test (Figure 1).

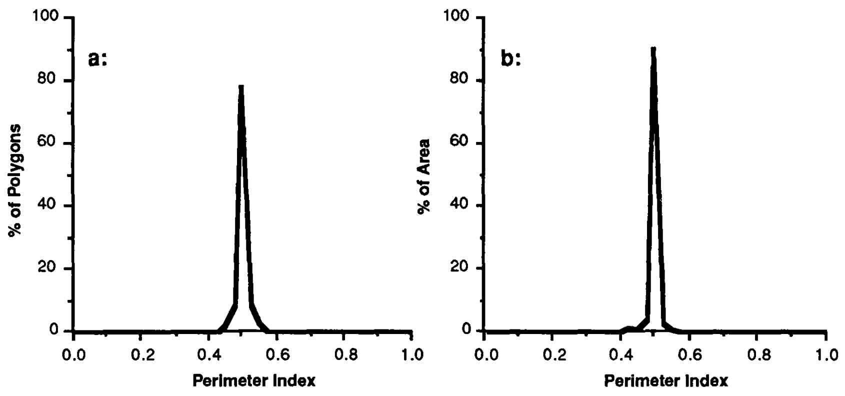
Figure 6: Attribute error (all lines from one source)



Even without a single source test, a perimeter index near zero or near one indicates an attribute-like error. However, while it is a more reliable classifier of error than the compactness index, larger polygons will occasionally, by chance, have a nearly equal proportion of perimeter from each source. Knowing where to draw the line between the ends and the positional error in the middle is not immediately obvious. One way to proceed is to introduce a known type of error, then see how the perimeter index varies.

To test the reliability of the index for classifying identification and discrimination error, positional error was introduced by translating (shifting) a map relative to itself. The original was used as the source of higher accuracy. Figure 7 shows the distribution of the perimeter index (for polygons of disagreement) after translating Map 1 south by a distance corresponding to .8 meters on the ground. This distance is quite small relative to the line width of the map, and an entirely possible registration error. These maps were digitized on a tablet with a least count resolution of about .4 meters on the ground at map scale. Variations in registration will arise from the hardware as well as the visual placement of the cursor over the registration marks.

Figure 7: Perimeter Index distribution from shifting Map 1 (distribution for all polygons (a) and area (b) in disagreement)

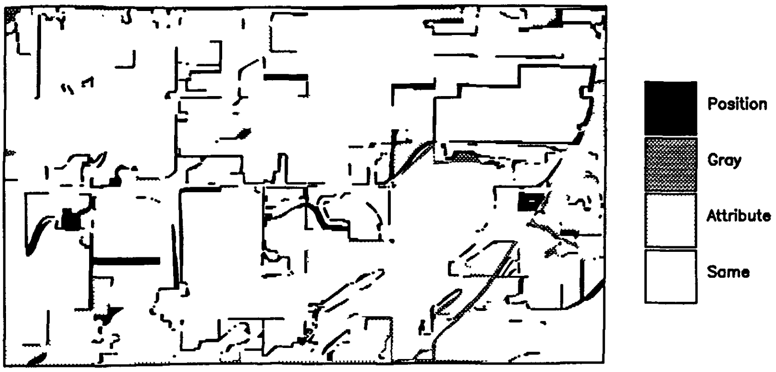


As Figure 7 shows, this purely positional error is closely packed around the theoretical value of 0.5. This indicates that the index correctly interprets the translation as a positional error.

Translation error, which would be due to misregistration or similar causes, is only one kind of positional error, but other forms of positional error also generate similar overall distributions. The result of the test of Map 1 on Map 2, shows a peak in Figure 4a much like the peak in Figure 7a, but the distribution by area in 4b is significantly different from 5b. When the restrictions are applied prior to applying the perimeter index, Figure 5a still shares the dominant central spike of 7a. Artificially pure positional error produces an unmistakable signature in the distribution of the perimeter index.

The distribution of the test results (Map 3, Table 1 and Figure 4) is the combination of all the error processes that distinguish the two sources. To decompose the polygons into two broad categories, position-like error and attribute-like error, they were analyzed for area, coincident boundary segments, compactness index and perimeter index. The compactness index is used to classify the more compact polygons, while the less compact polygons are classified with the perimeter index. While as yet there is no theory to guide the selection of a threshold between the medial and extreme values of the perimeter index, the range of values in Figure 7 and the breakpoints in Figures 4 and 5 indicate that from 0.4 to 0.6 is very likely to be positional error. Thus, for this application, any error with an index between 0.4 and 0.6 is termed positional, while those from 0 to 0.25 and from 0.75 to 1.0 are termed attribute-like. The remaining "gray" zone between 0.25 and 0.4 and between 0.6 and 0.75 (which contained 14 polygons and about 1% of the total area of the map) is ambiguous. In Map 4 it appears in a separate gray category. A symmetric set of thresholds is easily justified for a case, such a this one, of maps at the same approximate scale and level of detail. If one source has much more detailed lines, the midpoint might be biased due to the well-known effects of resolution on perimeter (Perkal, 1966).

Map 4: Positional and Attribute Errors



Map 4 shades the same polygons as Map 3, but classifies them into the position-like and the attribute-like categories. Again, the polygon borders are suppressed to show the small polygons. In general, the distinction of the two types seems reasonable, although there are some problems. A few large areas which were not due to uncertain boundaries were classified as position-like because they are classified as "narrow" by the compactness index and the perimeter contribution happened to be relatively balanced. Further refinement of the model may separate these from the more clearly position caused errors. A topological study of the chains may serve this purpose.

Table 2 shows the matrix crosstabulating the position-like error.

Table 2: Position-like Error (hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	—	22.8	.9	.8	34.0
Meadow	11.9	—	.04	.2	10.2
Coop Fld	.5	1.2	—	.03	1.3
Woods	.9	.3		—	1.4
Other	14.5	4.1	.1	.09	—

Table 3 shows the areas of polygons classified as attribute errors. In this situation, there are many fewer attribute errors, in terms of polygons, but the total area is greater. The nature of the attribute error should be interpreted in terms of the five categories. The largest attribute cell is 59.8 hectares which interpreter #1 classed as Row Crop and #2 classed as Meadow. This error will have little impact on the soil erosion plan. The errors involving the Other category will have greater impact, but are relatively small.

Table 3: Attribute Error (hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	—	59.8			15.2
Meadow	5.6	—			22.2
Coop Fld	3.5	2.4	—		.3
Woods				—	
Other	13.2	7.4		7.4	—

Attribute error can be decomposed directly into the cells of the table. An error between Meadow and Row Crop does not depend on the error between Woods and Other. Thus, any analysis of the attribute error relates to the particular classification and the ability to identify it on the source material.

In the spirit of a diagnostic test, there is a need to decompose more completely the error in Table 2, above. The positional error can have two systematic components which can be studied separately: bias in position – caused by misregistration, and filtering (or generalization). These two cases will be considered separately.

Translation

The experiment of translating a map against itself was described above in order to show a pure case of positional error. Such an experiment can be used as a diagnostic tool as well. By translating the map of higher accuracy, one may obtain a simulation of the amount of error which would arise from a given error. This distribution can be compared to the actual error discovered. Some goodness-of-fit procedure could be applied to pick out the best fitting translation. Error matrices are not random samples to which the usual tools of linear regression apply, but estimation tools from the toolkit of robust statistics would be the most applicable. The Least Median Square is one such method (Shyue, 1989).

Table 4: Misclassification Matrix
resulting from a .8 m South translation of Map 1 (figures in hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	1234.2	5.9	.4	.5	12.6
Meadow	7.2	256.9	.2	.2	2.1
Coop Field	.9	.2	40.9		.2
Woods	.7	.1	.03	12.6	.2
Other	11.3	3.1	.7	.3	250.4

For this test, no such fit was examined. The error polygons showed a set of narrow polygons elongated east-west (Map 3). These could have been generated by a north-south translation error. A small translation was performed, which produced the matrix shown in Table 4. The error of the translation was completely classified as position-like (see Figure 7). The distance of .8 meters was chosen as a reasonable (and small) number which produced an error matrix whose values nearly fit the actual errors discovered (see Table 2).

The translation error can be subtracted from the total positional error to produce a matrix of those errors which could not be ascribed to a simple registration problem (Table 5). A subtraction of the attribute error is not produced because translation produces only position-like error. In this case, the ambiguous error was aggregated with the position-like error, to judge the overall effect of translation. The subtraction does not remove the large instances of error, but it removes much of the small quantities from the matrix. In some cases, the translation produces a fraction of a hectare more than observed. Such small negative numbers are, in aggregate magnitude, less than the areas which they supplant, and should not detract from the use of translation in a larger model of error. It is not proven that any particular translation occurred, but such a test could assist in discovering the amount of a misregistration in a very specific manner. The residual error in Table 5 is not strictly proportional to the error reported in Table 2. For instance the errors involving Row Crop are all reduced, due to long boundaries, but Meadow/Other is much less affected.

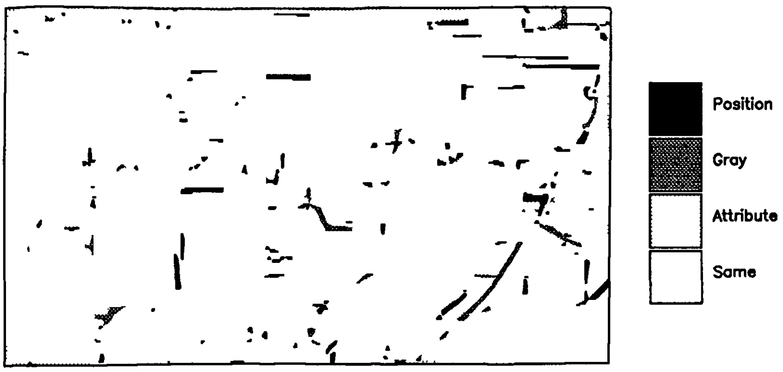
Table 5: Residual Positional Error [Table 2 minus Table 4] (hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	–	16.9	.5	.3	30.0
Meadow	4.7	–	-.2		11.5
Coop Field	-.4	1.0	–	.03	1.8
Woods	.2	.2	-.03	–	1.6
Other	8.2	1.0	-.6	-.2	–

Filtering

An alternative view of positional error does not seek a uniform translation, but it recognizes that maps are often digitized in greater resolution than is warranted by their accuracy. The overlay test in Map 3 and Table 2 was carried out at a very exacting tolerance (.4 m). This is essentially an exact overlay. Of course, it is critical to apply an exact test to determine the total error between two digital maps. But in addition, it is useful to study how much of the error observed comes from the classical slivers which are entirely unintentional (Goodchild, 1978). Cook (1983) included a distribution showing the areas of the objects, as shown indirectly in the graphs of Figures 2 &3. But the important characteristic of a sliver is narrowness, not area. A distance filter is much more appropriate. The epsilon filter (Dougenik, 1980; Chrisman, 1983; Beard, 1987) was applied during another overlay run. A series of tolerances were tried from 1 meter to 20 meters. The results of the test are reported from the most drastic filter, 20 meters. Map 5 presents the test results. This figure is a better estimate of the accuracy required for a land cover map for a soil erosion plan in this landscape. Any point on either map found within 20 meters of another point causes the cluster analysis of the WHIRLPOOL algorithm to ensure that only one will survive. This process does not average coordinates, it selects points.

Map 5: Test results after a 20 m filter



The first observation about Map 5 and Table 6, the misclassification produced by a 20 meter filtering overlay, is that the area in the diagonal increased from 85.8% to 88.4%. Thus, some area which was found to be in error with an exact overlay, was placed into the correct classifications if the positional tolerance was broadened to 20 meters. Some small error interactions discovered by the exact test disappear. Many categories are reduced substantially, while others are unaffected. This difference in behavior begins to discover the structure of error. The error below the 20 meter threshold may have essentially random distributions, but what survives may point at specific problems to correct.

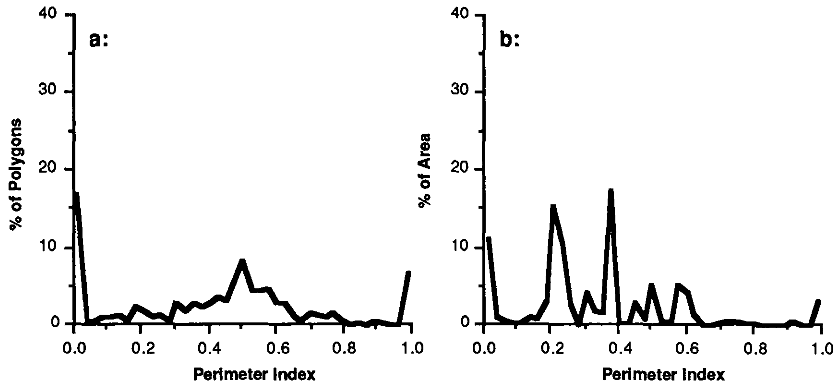
Table 6: Misclassification Matrix after filtering [20 m] (hectares)

Interp. 2:	Row Crops	Meadow	Coop Field	Woods	Other
Interp. 1:					
Row Crops	1136.6	76.5		.2	41.0
Meadow	12.1	221.6			32.2
Coop Fld	3.5	2.9	34.1		1.1
Woods	.04	.1		11.0	1.9
Other	24.1	8.9	.1	7.6	222.6

It is interesting to compare the effects of the 20 meter filtering and the .8 meter translation. The two figures seem to be radically different, but the error matrices are surpisingly similar. The difference is that the .8 meter translation occurs everywhere. All lines (at least those going east-west) generate slivers in proportion to their length. The 20 meter filtering actually moves things less. Compared to the exact overlay conducted in Map 3 which used 6048 points, the 20 meter filtering produces a representation in Map 5 with only 3241 points. It is a radical filtering, yet the basic message about the error between Map 1 and Map 2 is still there.

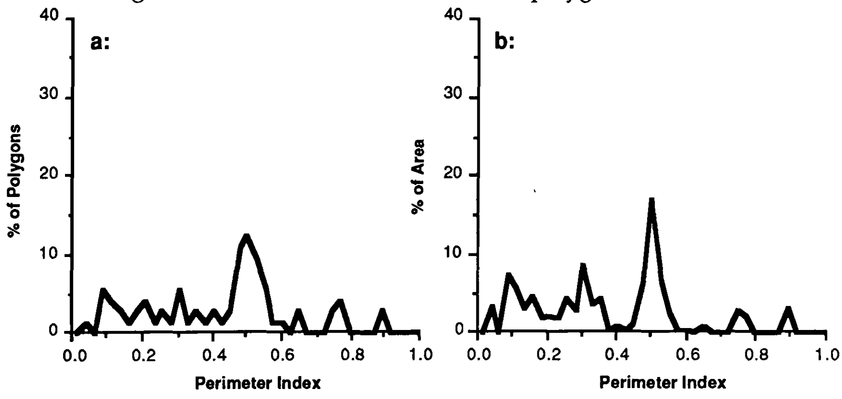
The filtering has an impact on the distribution of the perimeter index. Figure 8 shows the distribution for all polygons, as Figure 4 did for the exact case.

Figure 8: Distribution of perimeter index (all polygons 20 m filter)
a: by percentage of number of polygons; b: by percentage of area



There are many fewer polygons, but the distribution in 8a is well centered around 0.5. The filter removed many small slivers, reducing the numbers, and increased the discovery of coincident lines, making the distribution of indeces more balanced. Figure 9 shows the distribution of the tested polygons only, just as Figure 5 did above. Figure 9 shows a dramatic reduction in the positional (central) spike, though it still remains the mode of the distribution. Figure 9 involves many fewer polygons and less area than Figure 5.

Figure 9: Distribution of index (tested polygons, 20 m filter)



The filter has reduced the positional error from 6% to near 1% of the total area. Thus, we can infer that the positional accuracy of the features on these maps match to within 20 meters, except for 1% of more gross blunders. Such a statement could be refined by an iterative use of the filter.

Table 7 tabulates the errors reported in the positional category by the test at 20 meter tolerance. This is the positional error residual after the filtering. It seems to discover some of the same residual effects found by the translation. A few figures in this table (such as Row Crop/Meadow and Row Crop/Other) are roughly symmetrical around the diagonal, meaning that the one error was about as likely as the other. If the user is interested in overall figures for area, a finding of balanced error is similar to a finding of an unbiased estimator in statistics. However, if the user needs site-specific figures, the errors still are errors.

Table 7: Filtered Positional Error [20 m] (hectares)

Interp. 2: / Interp. 1:	Row Crops	Meadow	Coop Field	Woods	Other
Row Crops	—	6.3		.2	8.9
Meadow	4.4	—			2.2
Coop Fld	.004		—		.4
Woods	.04	.1		—	.4
Other	5.3	1.0	.1	.04	—

Some of the entries in Table 7 are not symmetric. These point out specific discrimination biases between the interpreters. This information, if produced as a test during a normal GIS production sequence would provide information beyond the typical accuracy assessment that would diagnose the specific pair of categories. Such information should enhance quality control.

The filtering procedure has less effect on attribute-like errors. The matrix of attribute error is presented in Table 8. The large propensity for interpreter 2 to see Meadow when #1 sees Row Crops carries over from the misclassification matrix. The size has risen from 59.8 ha. in Table 3 to 68.1. Most of the figures in this table have increased from Table 3. This matrix is notably less symmetrical compared to Table 7.

Table 8: Filtered Attribute Error [20 m] (hectares)

Interp. 2: / Interp. 1:	Row Crops	Meadow	Coop Field	Woods	Other
Row Crops	—	68.1			23.9
Meadow	5.6	—			27.9
Coop Fld	3.5	2.9	—		.6
Woods				—	1.5
Other	18.0	7.9		7.6	—

Considering the size of the map sheet, most of these attribute errors may be tolerable. Quality control efforts might apply to correct the positional errors as a higher priority, but some attention might also be given to Interpreter 2's propensity to classify Row Crops as Meadow.

Limitations

The test procedure developed in this paper is provisional. It does not classify all the errors entirely correctly. For example, there are some large errors declared to be position-like because the compactness index is relatively small and the perimeter index is balanced between the two sources, but the nature of the error seems to be much more a disagreement over classification, not position. These cases occur when a relatively compact polygon has an attached "tail" or when a polygon is a rectangle with a relatively narrow width (see the extreme left of Map 4). In these cases, unlike more typical position-like errors, the model will require further refinement.

In a more general sense, this test simply reports on the results for the area studied. It has no mechanism to estimate what would happen in some other, even nearby region. It does not have any particular statistical distribution or measure of goodness-of-fit. However, separation of the distinct forms of error is a first step towards the construction of such models.

Conclusions

This paper has attempted to demonstrate that a polygon overlay test is indeed possible and useful. It is possible to conduct a test using a replication of a map product, not necessarily a source of known higher accuracy. Differences of minimum mapping units and classification schemes are not a hinderance, but they are the very goal of a test. This test offers a chance to diagnose specific forms of mapping error. With some development and fine tuning it may come to replace the more standard point sampling methods used in the remote sensing discipline.

Acknowledgements
US National Science Foundation Grant SES 87-22084 provided partial support for this paper. The data used for the tests was generated by the Dane County Land Records Project during 1984 by Marcel Thoma and Kevin Connors. The University of Washington Statistical Consulting Service has provided a sounding board for some of the concepts, though the authors are responsible for any errors.

References Cited

Chrisman, N.R. 1982: Methods of spatial analysis based on error in categorical maps. PhD thesis, U. of Bristol.
Chrisman, N.R. 1983: The role of quality information in the long-term functioning of a GIS. *Proceedings AUTO-CARTO 6, 2,* 303-321.
Chrisman, N.R. 1987: The accuracy of map overlays: a reassessment. *Landscape and Urban Planning, 14,* 427-439.
Chrisman, N.R. 1989a: Modeling error in overlaid categorical maps. In: Goodchild and Gopal (ed.) *Accuracy of Spatial Databases,* London, Taylor and Francis: 21-34.
Chrisman, N.R. 1989b: Error in categorical maps: testing versus simulation. *Proceedings AUTO-CARTO 9,* 521-529.
Chrisman, N., Mezera, D., Moyer, D., Niemann, B., Vonderohe, A. 1984: Modernization of routine land records in Dane County, Wisconsin: implications to rural landscape assessment and planning, *URISA Professional Paper 84-1.*
Cook, B.G. 1983: Geographic overlay and data reliability, Proceedings IGU US/Australia Workshop: 64-80.
Douglas, D. 1974: It makes me so CROSS, Harvard Laboratory for

Computer Graphics, reprinted p. 303-307 in Peuquet and Marble 1990: *Introductory Readings in GIS,* Taylor & Francis.

Dougenik, J.A. 1980: WHIRLPOOL: A geometric processor for polygon coverage data, *Proceedings AUTO–CARTO IV, 2:* 304-311.

Fitzpatrick-Lins, K. 1978: Comparison of sampling procedures and data analysis for a land use and land cover map, *Photogrammetric Engineering and Remote Sensing, 47,* 343-351.

Goodchild, M. 1978: Statistical aspects of the polygon overlay problem, in vol. 6 G. Dutton, ed., *Harvard Papers on Geographic Information Systems,* Addison Wesley.

Mark, D.M. and Csillag, F. 1989: The Nature of Boundaries in 'Area-Class Maps', *Cartographica, 26*(1), 65-78.

Morrison, J. (ed.) 1988: The Proposed Standard for Digital Cartographic Data, *The American Cartographer,* **15**: 9-140.

Niemann, B. Sullivan, J. Ventura S., Chrisman, N. Vonderohe, A. Mezera, D. and Moyer, D. 1987: Results of the Dane County Land Records Project, *Photogrammetric Engineering and Remote Sensing, 53,* 1371-1378.

Perkal, J. 1966: An attempt at objective generalization. *Discussion Paper 10,* Michigan InterUniversity Community of Mathematical Geographers.

Rosenfield, G. and Melley, M. 1980: Applications of statistics to thematic mapping. *Photogrammetric Engineering and Remote Sensing, 46,* 1287–1294.

Shyue, S-W. 1989: High breakdown point robust estimation for outlier detection in photogrammetry, PhD dissertation, University of Washington.

Sinton, D. 1978: The inherent structure of information as a constraint to analysis: mapped thematic data as a case study , in vol. 7 G. Dutton, ed., *Harvard Papers on Geographic Information Systems,* Addison Wesley.

Unwin, D. 1981: *Introductory Spatial Analysis,* Methuen, London.

Ventura, S. 1988: *Dane County Soil Erosion Plan,* Land Conservation Committee, Madison WI.

Ventura, S. Sullivan, J.G. and Chrisman, N. 1986: Vectorization of Landsat TM land cover classification data. *Proceedings URISA, 1,* 129-140.

Vonderohe, A.P. and Chrisman, N.R. 1985: Tests to establish the quality of digital cartographic data: some examples from the Dane County Land Records Project, *Proceedings AUTO–CARTO 7,* 552-559.

GéoGraph: A Topological Storage Model for Extensible GIS

K. Bennis (1), B. David (2), I. Morize-Quilio (1), J.M. Thévenin (3), Y. Viémont (1)

(1) MASI, Université Paris VI
45, Avenue des Etats Unis, 78000 Versailles. France
e-mail: viemont@sabre.ibp.fr

(2) IGN-France
2, Avenue Pasteur, BP 68, 94160 Saint Mandé.France
e-mail: david@ign.uucp

(3) INRIA-Rocquencourt
BP 105, 78153 Le Chesnay.France
e-mail: theven@madonna.inria.fr

Abstract

Topological data structures are useful for reducing the cost of geometrical operations within a Geographic Information System (GIS). Unfortunately, manipulating such data structures can be quite complex -- especially when supporting multiple, overlapping geographical maps. The GéoGraph storage model proposed in this paper solves this problem. It is implemented as a toolbox, and is used as a low-level system layer for support of a GIS. The GéoGraph storage model is based on a graph with the corresponding basic traversal primitives, and can be integrated within an extensible relational DBMS so that important spatial operations can be directly executed by graph traversals. Furthermore, the graph is decomposable so that only the useful subset of the database can be loaded from disk without format conversion.

1. Introduction

Geographic information systems (GIS) require storage and manipulation of both semantic and spatial data. Whereas conventional DBMS data models (e.g., the relational model) are well suited to representing and manipulating semantic data, queries concerned with spatial data imply the use of geometric operations not directly supported by these data models. Furthermore, because the processing performance of geometric operations is strongly influenced by data representation, systems supporting spatial data benefit greatly from a data model specially tailored for efficient support of these operations.

There are several ways to implement spatial data. A simple solution is to store each spatial object as a coordinate list. Although coordinate lists reflecting the position of objects are sufficient to perform geometric operations, inter-object spatial relationships that are obvious when seen on a map are complex and costly to capture when using this representation. To reduce the number of inter-object comparisons required by this approach, it is possible to use spatial indices on coordinate lists. As a more fundamental attack on this problem, however, it is possible to enrich the geometrical description of objects with topological information. This information explicitly materializes the connectivity and contiguity relationships between spatial objects, and can be represented as a graph that provides direct and efficient support for adjacency operations.

Topological information has been used in several geographical information systems [Morehouse85, Herring87, Kinnear87, Spooner90]. The topological information is usually stored in a graph using an appropriate internal representation [White79, Peuquet84]. These representations can be complex to maintain and may require expensive verification of integrity constraints. In many cases, topological graphs have been used to store relationships between spatial objects of one geographical map at a time. For instance the topology of a road map is stored separatly from the topology of a land cover map. As a consequence operations involving several maps require the fusion of several graphs, which can be a complex and expensive procedure [Schaller87].

In this paper we present GéoGraph, a storage model for topological information that supports efficient operations involving several layers of maps. This model stores the topology of an internal map corresponding to the overlay of several geographical maps. Hence spatial objects of one geographical map are decomposed into collections of elementary spatial objects and the internal map materializes the relationships between the spatial objects. This principle has alrady been used in some GIS like TIGER [Meixler85] and TIGRIS [Herring90]. We focuse on a clean integration of topological information in a DBMS so that semantic and spatial data can be manipulated in a uniform way.

GéoGraph is implemented using a toolbox approach and constitutes a low-level system layer that can support general purpose GIS. This storage model is based on the topological map theory that guarantees coherent updates of topological information, and provides a minimal set of operations for navigation through a topology [Dufourd88]. In this paper, we demonstrate a straightforward integration with an extensible relational DBMS supporting a GIS. The integration is based on a single graph that incorporates both relational data and spatial data in order to precompute all operations, and is currently being investigated in the framework of the GéoTropics system [Bennis90], an extensible DBMS based on extensions of SQL.

The paper is organized as follows. Section 2 reviews the basic concepts of topological maps used in GéoGraph. Section 3 introduces the concept of map overlay and then provides a formal definition of GéoGraph in terms of a graph structure and primitive operations on the graph. Section 4 illustrates the use of the GéoGraph toolbox for implementation of geographical operators of the GéoTropics system. Section 5 argues for a specific implementation of GéoGraph when it is incorporated into a relational DBMS. Section 6 gives our conclusions.

2. Topological map concepts

Topological maps have been defined as an extension of combinatorial maps [Edmond60, Cori81]. They provide the necessary support for expressing relationships between spatial objects in a plan [Dufourd88, Dufourd89]. This section gives intuitive definitions of the basic concepts of topological maps in order to highlight their contribution for topological information management in cartography. To clarify the discussion we distinguish non fully-connected topological maps from topological maps.

2.1. Non fully-connected topological maps

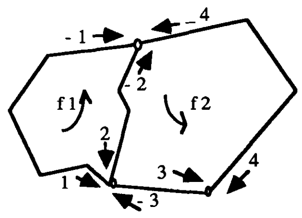
A non fully-connected topological map defines a graph similar to the well known topological graph [White79, Peuquet84], and uses two basic functions -- α and σ (see figure 1). In general this graph is not fully-connected. Edges of this graph represent lines which correspond to the location of linear features (e.g., roads), or boundaries of surfacic features that we call faces. Nodes of this graph represent intersections of edges. Conceptually, each edge is decomposed into two blades labeled by integers (e.g., b and -b) corresponding to the two possible orientations of the edge. Function α applied to a blade label gives the label corresponding to the opposite blade orientation. Function σ is a permutation which orders blades around their end-node in a clockwise fashion. Thus, σ applied to blade b gives the next blade ending at the end-node of b. Any traversal of the graph can be expressed by a combination of the functions α and σ. For instance, the boundaries of one face can be traversed turning counterclockwise applying a loop on function $\varphi = \alpha \circ \sigma$ to any blade of that face. In the graph, a geometry is associated to each edge. For this purpose, a last function γ is defined such that γ applied to a blade gives a coordinate list corresponding to the geometry of the associated edge. A non fully-connected topological map can be defined more formally as follows:

Definition: non fully-connected topological map

A non fully-connected topological map is defined as a quadruplet $(B, \alpha, \sigma, \gamma)$ where B is a finite set of blades; $\alpha : B \to B$ is a permutation such that $\forall b \in B\ \alpha(b) = -b$; $\sigma : B \to B$ is a permutation such that $\forall b \in B\ \sigma(b)$ is the next blade ending at the end node of b, turning clockwise; and γ is a function which applied to any blade returns the geometry of the associate edge.

A permutation φ can be deduced from α and σ such that $\varphi = \alpha \circ \sigma$.

According to this definition, the cycles $(b,\alpha(b))$ of α define edges, the cycles $(b,\sigma(b),\sigma \circ \sigma(b),\ldots,\sigma^{-1}(b))$ define nodes and the cycles $(b,\varphi(b),\varphi \circ \varphi(b),\ldots,\varphi^{-1}(b))$ of φ define faces. It is important to note that any blade defines one and only one edge, node, and face using respectively a cycle of α, σ or φ. The reverse is not true. From these properties we can deduce that, given a face defined by blade b and function φ, the adjacent area along blade b is defined by blade $\alpha(b)$ and function φ.



Edges:	Nodes :	Faces :
e1 = (1,-1)	n1 = (1, 2, -3)	f1 = (1, -2)
e2 = (2,-2)	n2 = (-1, -4, -2)	f2 = (2, 3, -4)
e3 = (3,-3)	n3 = (3, 4)	
e4 = (4,-4)		

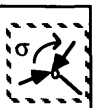


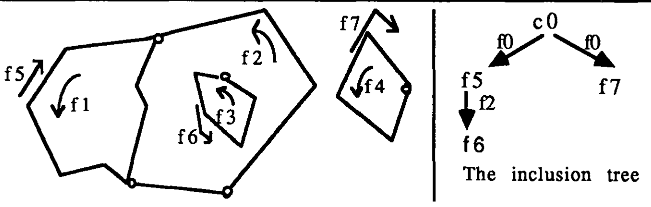
Figure 1: An example of map

Compared to the usual adjacency graphs, non fully-connected topological maps have the following advantages for geographic applications: (i) faces are easily enumerated; (ii) the planarity of a map can be checked very efficiently using α, σ, φ and γ [Dufourd89].

2.2. Topological maps

Applying the definition of a non fully-connected topological map, the graph resulting from a geographical map is in general not fully-connected, and useful relationships between different fully-connected components of this graph are not captured. In order to avoid this drawback, a topological map is defined below as an extension of a non fully-connected topological map, where a partial order among the fully-connected components is defined based on *geometric inclusion*. In support of this definition, we note that given any pair of fully-connected components (c1, c2), the following property is true: either c1 and c2 locations are disjoint (side by side), or one of c1 or c2 is fully included into one face of the other component. Otherwise (c1, c2) would form a single fully-connected component. In the second case, one component, say c1, constitutes a hole in a face of the other component (c2).

A hole is an external face which is defined by the external boundaries of a connected component. In order to distinguish internal faces from external faces a convention is introduced: an internal face is defined by its boundaries, turning counterclockwise; an external face is defined by its boundaries, turning clockwise. Figure 2 shows the graphic representation of a topological map.



f5 and f7 are the most external faces. f6 is included into the internal face f2 of the connected-component f5

Figure 2: an example of topological map

A topological map can be defined formally as follows:

Definition: topological map

A topological map is defined by: (i) a quadruplet (B, α, σ, γ) which is a non fully-connected topological map; (ii) an inclusion tree τ, composed of nodes c_i representing the fully-connected components of the quadruplet (B, α, σ, γ) and arcs ($c_1 \rightarrow c_2$) connecting two components c_1 and c_2 iff c_2 is included in c_1. An arc ($c_1 \rightarrow c_2$) is labeled by the face of c_1 containing c_2. The root of τ is a virtual component c_0 containing all the components of the map.

3. The GéoGraph model

A topological map can efficiently handle all adjacency operations between objects of a map, but is restricted to operations applied to objects belonging to the same map. In cartography, however, the same area is often represented through several maps, each map representing spatial objects associated to a particular semantic point of view (e.g., road map, land cover, etc.), and users frequently apply complex operations involving several such maps. The GéoGraph model was therefore designed to efficiently deal with a topology involving several layers of maps.

The purpose of this section is to present an extension of topological maps supporting map overlay, and to then define the GéoGraph model. Our model is expressed in terms of a specific graph structure called GéoGraph (which represents an extended topological map), plus a set of basic traversal primitives for GéoGraphs.

3.1. Extending topological maps

Operations involving several layers of maps are based on costly geometrical intersection computations. Optimized computational geometry algorithms for computing shape intersections have been studied [Preparata85], but, even with the use of supporting index structures, they remain slow.

To avoid geometric intersection computations during query processing, intersections between spatial objects (regions, lines and points) of several (overlapping) maps can be pre-computed during the creation of the database. This technique has been exploited in the GEOQL system [Sack87], where geometric processing is reduced by adding to the object coordinate lists of several maps the points corresponding to inter-map object intersections. Checking for object intersections then consists of looking for explicit shared points.

The GéoGraph model is based on pre-computing a collection of elementary spatial objects (ESOs) that correspond to a decomposition of the spatial objects of the original maps. Figure 3 illustrates the result of this pre-computation step applied to the overlay of a map representing Paris's districts and a map representing the underground rail network of Paris. The ESO collection resulting from the pre-computation step constitutes a new map which can be stored as a topological map where the ESOs are represented as faces, edges, and nodes. To speed up operations defined on the original maps while actually using the ESO map in computations, it is necessary to keep links between the original objects and the ESOs. These links materialize aggregations of ESOs which represent original elements. On the example of figure 3, district number 8 has been split in two faces which have to be aggregated to reconstruct the map of Paris's districts. Topological maps can be extended to maintain these aggregation links.

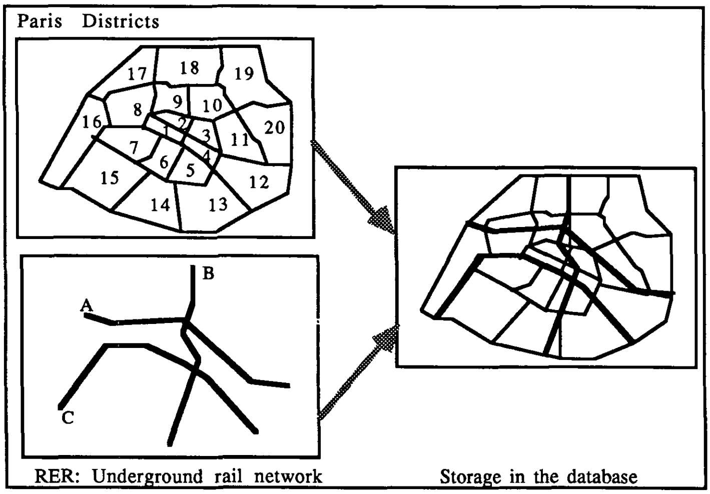


Figure 3: The corresponding storage of two themes.

A topological map extended with aggregation links maintains adjacency relationships between ESOs as usual, plus intersection and inclusion relationships between aggregations of ESOs. Using an extended topological map, most of the operators involving intersection and inclusion relationships on several maps can be evaluated using graph-based processing instead of geometric processing.

3.2. GéoGraph definition

We first introduce a few notations appropriate in the context of a collection of geographical maps. In most of our examples M will denote a particular geographical map. A geographical map M is described by semantic data and spatial data. The spatial data of a geographical map are called regions, lines, or points, and represent, respectively, surfacic features, linear features or ponctual features. We denote by R, L, or P, sets of regions, lines, and points, and by r, l, or p, the elements of these sets. The spatial data of a particular geographical map constitute a subset of R, L or P which is identified by a unique name. We use S to denote one of these subsets. The ESO resulting from the pre-computation step applied to the collection of geographical maps are called faces (inner faces), holes (outer faces), blades and nodes. We denote by F, H, B, N, or C, respectively, the sets of faces, holes, blades, nodes and coordinate lists, and by f, h, b, n, or c, the elements of these sets. A region is an aggregation of faces and a line is an aggregation of blades. A point is associated to a node.

The GéoGraph graph is a specitic representation of the extended topological map described in section 3.1. This graph is illustrated in figure 4. Nodes of this graph are elements of R, L, P, F, B, and N. Nodes and faces are explicitly represented in this graph (unlike the representation of topological maps given in section 2) in order to more directly associate spatial data with semantic data in the database. Edges of the GéoGraph represent three kinds of functions: functions connecting elements of the original maps to ESOs; functions on the topological maps; and functions connecting each blade to the nodes representing the face and the node defined by the blade. Edge functions connecting elements of an original map to ESOs are identified using the name associated by the map to these elements. This allows retrieving from ESOs the original elements of a particular map. The graph of GéoGraph can be defined more formally as follows.

Definition:

GéoGraph is a graph (X, A) where $X = R \cup L \cup P \cup F \cup H \cup B \cup N \cup C$ is a set of vertices of G and A is the set of edges defined below:

(i) Links between spatial objects and ESO:

- $(r, S, f) \in A$ iff $r \in S$, $S \subset R$, $f \in F$ and f is a component of r,
- $(l, S, b) \in A$ iff $l \in S$, $S \subset L$, $b \in B$ and b is a component of l,
- $(p, S, n) \in A$ iff $p \in S$, $S \subset P$, $n \in N$ and n correspond to p,

(ii) Topological links:

- $(b, \alpha, b') \in A$ iff $b \in B$, $b' \in B$ and $\alpha(b) = b'$,
- $(b, \sigma, b') \in A$ iff $b \in B$, $b' \in B$ and $\sigma(b) = b'$,
- $(b, \varphi, b') \in A$ iff $b \in B$, $b' \in B$ and $\varphi(b) = b'$,
- $(b, \gamma, c) \in A$ iff $b \in B$, $c \in C$ and c is the coordinate list associated with b,
- $(f, h) \in A$ iff $f \in F$, $h \in H$ and h is a hole into the internal face f,

(iii) Correspondence between faces, nodes and blades:

- $(b, f) \in A$ iff $b \in B$, $f \in F$ and f is the left face of b,
- $(b, h) \in A$ iff $b \in B$, $h \in H$ and h is the left hole of b,
- $(b, n) \in A$ iff $b \in B$, $n \in N$ and n is the end node of b.

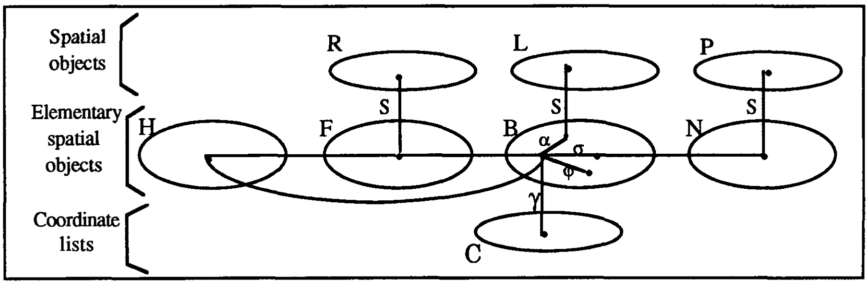


Figure 4: Links between objects in GéoGraph

3.3. Primitive operations

The GéoGraph model provides a set of basic primitives to traverse a GéoGraph. These primitives constitute a toolbox dedicated to the implementation of efficient geometric operations involving several maps via graph traversal operations. This set of primitives is detailed below:

- traversals of aggregation links between ESO and regions, lines or points are supported by the following primitives :
  - (i) MemberFaces (r, S) = $\{f \in F / r \in R$ and $(r, S, f) \in A\}$,
  - (ii) OwnerRegions (f, S) = $\{r \in R / f \in F$ and $(r, S, f) \in A\}$,
  - (iii) MemberBlades (l, S), OwnerLines (b, S), MemberNode (p, S), Ownerpoint (n, S) are defined similarly. Note that MemberNode and Ownerpoint are singletons;
- traversals of the topological map defined on ESO are supported by :
  - (i) α, σ and φ which are the conventional permutations of topological maps,
  - (ii) γ gives the coordinate list,
  - (iii) The primitives necessary to traverse the inclusion tree of topological maps :
    ContainingFace (h) = $\{f \in F / h \in H$ and $(f, h) \in A\}$,
    ContainedFaces (f) = $\{h \in H / f \in F$ and $(f, h) \in A\}$;
- traversals between faces holes and nodes defined by blades:
  (since faces and nodes are described by a list of blades, the inter-connections of blades, faces and nodes are expressed in the next primitives)
  - (i) Left (b) = $\{f \in F / b \in B$ and $(b, f) \in A\}$ (Left (b) is a singleton),
    BoundingBlades (f) = $\{b \in B / f \in F$ and $(b, f) \in A\}$,
  - (ii) Left (b) = $\{h \in H / b \in B$ and $(b, h) \in A\}$ (Left (b) is a singleton),
    BoundingBlades (h) = $\{b \in B / h \in H$ and $(b, h) \in A\}$,
  - (iii) EndNode (b) = $\{n \in N / b \in B$ and $(b, n) \in A\}$ (EndNode (b) is a singleton),
    ArrivingBlades (n) = $\{b \in B / n \in N$ and $(b, n) \in A\}$.

GéoGraph primitives allow traversal of all edges of a GéoGraph in both directions, so any traversal of a GéoGraph can be expressed by a combination of these primitives. Within a GéoGraph, adjacency relationships between objects of sèveral geographical maps can be deduced from traversals along aggregation edges and along edges materializing the topological map defined on the ESOs. Containment relationships between objects of the same type (region/region or line/line) belonging to different geographical maps can be deduced directly from traversals along aggregation edges. Containment relationships between objects of different types (region/line, region/point or line/point) belonging to different geographical maps can be deduced from traversals along aggregation edges and along edges materializing the topological map defined on the ESO. GéoGraph primitives are thus sufficient to carry out all the operations based on the inter-object relationships.

To ease the implementation of these operations it is possible to deduce functions useful as construction blocks in the design of a GIS. For example, the function AdjFaces(f) = {Left (b) / b ∈ BoundingBlades (f)} can be defined to retrieve the adjacent faces of face f; the function Right(b) = Left (α(b)) can be defined to retrieve the right face of blade b. In summary, GéoGraph supports all the links between spatial objects which are important to efficient geometric operators. We leave unspecified the details concerning links between spatial objects and semantical objects. These links are not included in the GéoGraph model for they depend on the data model used in the GIS.

4. Using the GéoGraph model

This section illustrates the implementation of spatial operators as an extention of a DBMS using the GéoGraph model. For the sake of clarity, the spatial operators are presented in a relational context, but it is important to note that these operators and their implementation can be generalized to other data models.

First, we present a possible integration of the GéoGraph data in a database using a relational DBMS which can be extended with abstract data types [Stonebraker83]. Then, classical spatial predicates and spatial functions are enumerated. Spatial predicates are applied to a couple of spatial objects to check some spatial properties. The definition of the main spatial operators required in a relational DBMS extended toward geography is then introduced. These operators are based on spatial predicates and spatial functions. They work on sets of spatial objects and return the combination of spatial objects which satisfy a given predicate. Finally the implementation of these operators with the GéoGraph model is detailed. The architecture and the query langage of such a DBMS extended toward geography can be found in [Bennis90].

4.1. Connection of GéoGraph with an extended relational database

A cartographic object is composed of semantic and spatial data. In order to support the spatial data, the spatial domains Regions, Lines and Points are added to the conventional domains of values used in relational DBMSs. A spatial relation is then defined as a relation containing at least one attribute which takes its values in a spatial domain. A map is a spatial relation with exactly one spatial attribute which is a key of the relation. The conventional relational operators (i.e. selection, join and projection) are augmented with spatial operators for spatial attribute manipulations.

To clarify the discussion, a few notations are introduced below. We use two spatial relations named R and S. We denote by R.k (resp S.l) the spatial attribute of R (resp S). We denote by r (resp s) a tuple of the R (resp S) spatial relation. We denote by ΔV a set of values varying on the same domain and by v a particular value of this set. The result of a spatial operation is a relation denoted by RES.

In the sequel of the paper we assume for clarity that the database is stored according to the DBGraph storage model introduced in [Pucheral90]. This model stores a relational database as a bipartite graph composed of a set of tuples named T, a set of values named V and edges connecting each tuple to each of its attribute values (see figure 5). The purpose of this storage model is to precompute all conventinal relational operations, which is complementary with the objective of the GéoGraph. Two basic traversal primitives are provided: succ_tup(t, R.k) is a function from T to V that delivers the value of attribute R.k of tuple t and succ_val(v, R.k) is a function from V to T that delivers the set of tuples whose th attribute value is v.

In a DBGraph, tuple vertices (resp. value vertices) may be grouped on a relation basis (resp. domain basis) since the relations form a partition of T (resp. V). Spatial domains constitute three subsets of V which comprise the sets R, L, P used as entry points in the GéoGraph.

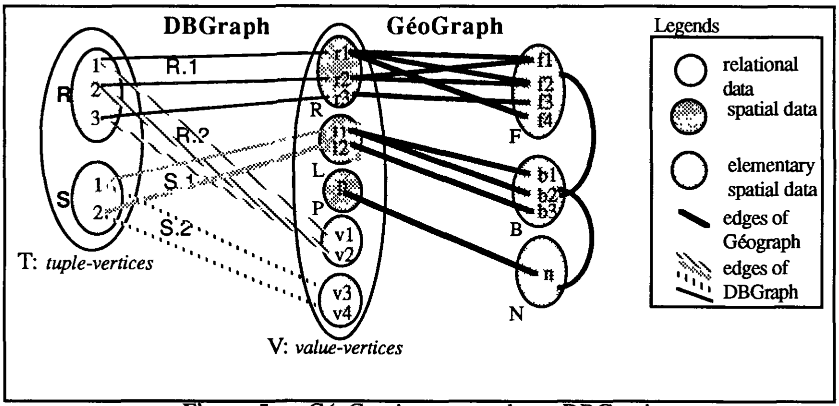


Figure 5: a GéoGraph connected to a DBGraph

4.2. Spatial Predicates and Spatial Functions.

A spatial predicate takes two input spatial attribute values, and checks whether a given spatial property is satisfied by this pair. The two input values can be of different domains, although there are some restrictions depending on the predicate. Figure 6 summarizes the classical spatial predicates and the domains of the allowed input values. Spatial predicates fall into two categories: one for checking neighborhood relationships such as adjacency of regions; the other for checking containment relationships such as inclusion or overlap of spatial objects. For detailed information on these predicates see [David89]. Each spatial predicate can be evaluated by a traversal of the GéoGraph. This traversal can be expressed by a sequence of primitive operations of the GéoGraph model. One part of this traversal corresponds to a translation of the regions, lines or points given in entry into ESO (faces, blades and nodes), while the other part selects elementary components which satisfy the predicate. The first part uses operations on aggregation links and the second part uses operations of topological maps. Examples of such traversals can be found in section 4.4.

Predicate	Region	Line
Region	Adjacent Overlap Inclusion	Border Overlap
Line	Right Left Overlap Inclusion	Connected Overlap Inclusion
Point	Inclusion	Ends Inclusion

Figure 6: The spatial predicates

Spatial functions are useful to calculate new values which are either numerical or geometrical (coordinates lists). Some are also used to compute new values of type region, line or point based on new aggregations of ESO. These functions can be classified into two categories: those requiring a geometrical computation on spatial values, and those which can be sped up by the topological map included in the GéoGraph model. Figure 7 summarizes the main spatial functions.

Functions	Unary	Binary
Speed up by GéoGraph		**Result ∈ R or L or P** Intersection(O1,O2) with O1 ∈ R or L and O2 ∈ R or L Fusion(O1,O2) with O1 ∈ R andO2 ∈ R or O1 ∈ L andO2 ∈ L Difference(O1,O2) with O1 ∈ R andO2 ∈ R or O1 ∈ LandO2 ∈ L
Involve a geometrical computation	**Result is numerical** Area(O1) with O1 ∈ R Perimeter(O1) with O1 ∈ R Length(O1) with O1∈ L **Result is a coordinate list** Geometry(O1) with O1 ∈ R or L	**Result is numerical** Distance(O1,O2) with O1 ∈ R or L or P and O2∈ R or L or P

Figure 7: The spatial functions

4.3. Spatial Operators

The data model of GéoTropics uses three basic spatial operators: the spatial selection, the spatial join, and the calculation operators. The first two of these correspond to extensions of the conventional selection and join operations of the relational model. The spatial operators take as arguments one or two spatial relations and return a spatial relation, and are now explained

The **Spatial Selection** operator, denoted by Sel(S, Q), is applied to a relation S, and determines the subset ΔS of the tuples of Swhose spatial attribute satisfy a qualification Q. The qualification Q is a simple comparison S.l Θ const where const is a spatial constant of the domain region or point and where Θ is a spatial predicate. The spatial predicate used is generally inclusion or overlap. Selections involving these predicates correspond to the usual geographical operations of clipping and windowing. The selection operator is expressed as:

$\mathrm{Sel}(S, Q) = \{s \in S / (s.l \ \Theta \ c) \text{ is true}\}$

The **Spatial Join** operator denoted by Join(R, S, Θ), is applied to the spatial relations R and S, and determines a set of tuples composed of all possible combinations of a tuple $r \in R$ concatenated to a tuple $s \in S$, such that the spatial attributes R.k and S.l of r and s satisfy the join condition R.kΘS.l, where Θ is a spatial predicate. Most of the spatial operations based on relationships between spatial objects can be expressed by a join operation involving a specific spatial predicate. The join operator is expressed as follows:
$\mathrm{Join}\ (R, S, \Theta) = \{(r, s) / r \in R, s \in S \text{ and } (r.k \ \Theta \ s.l) \text{ is true}\}$

The **Spatial Calculation** operator, denoted by Calc (R, f) or Calc (R, S, f) can take one or two relations as arguments. It is similar to the spatial join, but extends the concatenation of the entry relations attributes with the result of a spatial function f applied to the spatial attributes. Note that the join predicate is replaced by function f. This operation can be expressed as follows :
$\mathrm{Calc}\ (R, f) = \{ (r, f(r.k) / r \in R \}$
$\mathrm{Calc}\ (R, S, f) = \{ (r, s, f(r.k, s.l)) / r \in R, s \in S\}$
The widely used overlay operation can be translated as follows: projection (Calc (R, S, ∩)) where ∩ involve R.k, S.l and the projection discards these two attributes.

4.4. Spatial Operator Implementation.

The GéoGraph model is currently being used to support the three spatial set-oriented operators presented in section 4.3. Numerous versions of these operators can be deduced depending on the spatial predicate used. It is not possible to give three general algorithms that support all versions of these operators. For illustration, this section focuses on two versions of the join operator and one version of the calculation operator which correspond to the more commonly used geometric operations and illustrate well the functionnality of the GéoGraph model. The join operation is first applied to retrieve all couples of adjacent regions of two maps (spatial relations). Then it is applied to retrieve all couples of overlaping regions and lines of two maps. The joins involves the adjacency and the containment relationships, which belong to the two classes of spatial predicates. The classical operation of overlay is then expressed in primitives of the GéoGraph model. The execution of the selection operator is not detailed in this section because it is efficiently handled by algorithms involving geometrical indices. For this operation the GéoGraph model had to be augmented with geometric indices for elementary components (Face, Blade and Node) and spatial attribute values (region, line or point) (see section 5).

Let us consider the adjacent-join, a **spatial join** operation involving an adjacent predicate. Two regions are adjacent if they have adjacent faces and if they do not have common faces. The algorithm performing the operation is given figure 8. This algorithm may be decomposed into four steps, each of which is a traversal of a subpart of the GéoGraph. For each tuple r of R, the first step decomposes the region-value of attribute R.k into faces. The second step executes a sequence of topological operations on ESOs in order to get the set of faces which are adjacent to some faces obtained in the first step. The third step converts the faces obtained in step2 into region-value of attribute S.l. All these regions contain at least a face adjacent to one face of the region-value of r. The fourth step checks that region-values selected in the 3rd step do not have common faces with the region-value of r.

```
Function Join (R, S, Adjacent)
/* R.k and S.l take their values in R */
/* we assume card(R) < card(S) */
begin
/*....................................................................1st step */
for each r ∈ R do
    ΔV3 = ∅;
    ΔV1 := MemberFaces(r.k, R.k);          /* decomposition into elementary faces */
    /*................................................................2ndstep */
    for each v1 ∈ ΔV1
        ΔV2 := BoundingBlades (v1);                  /* give all blades bounding v1 */
        for each v2 ∈ ΔV2
            ΔV3 := ΔV3 ∪ Left (v2);                   /* give adjacent faces of v1 */
        endfor
    endfor
    ΔV3 := ΔV3 - ΔV1;                          /* discard faces belonging to r.k */
    /*................................................................3rd step */
    for each v3 ∈ ΔV3
        ΔV4:=ΔV4 ∪ OwnerRegions(v3, S.l);   /* retrieve regions of S.l including face v3 */
    endfor
    /*................................................................4th step */
    for each v4 ∈ ΔV4
        if ΔV1 ∩ MemberFaces(v4, S.l) ≠ ∅         /* discard  the regions which have */
                                                  /* common faces with region r.k */
        then RES := RES + (r , succ_val(v4, S.l) );   /* build the result made of tuple r */
                                                  /* and the tuple owning the region v4 */
endfor
end
```

Figure 8: Join operator involving the adjacency predicate

Consider now the overlap-join operator, a join operation involving a spatial predicate checking the overlap between a line and a region. The algorithm performing this operation is given figure 9 which starts from the lines to reach the overlaping regions. A line and a region overlap if at least one blade of the line defines two faces belonging to the region. This algorithm is also based on four steps which are similar to the four steps of the previous algorithm. Nevertheless, these steps cannot be expressed in exactly the same way. The first three steps determine the set of regions on the left of the line and the set of regions on the right of the line. The fourth step performs the intersection of these two sets.

```
Function Join (R, S, Overlap)
/* R.k takes its value in L
and S.l takes its value in R */
/* we assume card(R) < card(S) */
begin
/*.................................................................1st step */
for each r ∈ R do
    $\Delta V_4 := \emptyset$;
    $\Delta V_1$ := MemberBlades(r.k, R.k)        /* decomposition into elementary blades */
    for each $v_1 \in \Delta V_1$
        /*.........................................................2nd step */
        $v_l$= Left($v_1$);                          /* give the face on the left of blade $v_1$ */
        $v_r$:= Left($\alpha(v_1)$);                 /* give the face on the right of blade $v_1$ */
        /*.........................................................3rd step */
        $\Delta V_2$ := OwnerRegions($v_r$, S.l);    /* retrieve regions of S.l including face $V_r$ */
        $\Delta V_3$:= OwnerRegions($v_l$, S.l);     /* retrieve regions of S.l including face $V_l$ */
        /*.........................................................4th step */
        $\Delta V_4 := \Delta V_4 \cup (\Delta V_2 \cap \Delta V_3)$;   /* discard the regions which border line r.k */
    endfor
    for each $v_4 \in \Delta V_4$
        RES = RES $\cup$ (r , succ_val($v_4$, S.l) );   /* build the result made of tuple r */
                                                      /* and the tuple owning the region $v_4$ */
endfor
end
```

Figure 9: Join operator involving the overlap predicate

Finally, let us consider a version of the spatial **Calculation operator** involving the function of intersection of two regions. This operation is applied to two maps and produces the overlay of the maps. The corresponding algorithm is given figure 10. If we compare this algorithm to the previous ones, it can be decomposed into four steps with an empty second step, since containment relatioships between objects of the same type can be deduced from traversals along aggregation edges without the use of the topology. In this algorithm, the computation of $\cap$ (r.k, s.l) is reduced to a simple set-intersection between two sets of faces with no access to the coordinates of the spatial objects geometry .

In our experience, most of the algorithms of binary geometrical operations can be decomposed into four steps. The first step decomposes the spatial attribute values of the first map into a set ESO1, using aggregation edges. The second step performs topological operations starting from ESO1 to reach a set ESO2 satisfying a topological predicate. The third step retrieves spatial attribute values of the second map which aggregate elements of ESO2, and the fourth step performs some supplementary verifications. For operations involving containment relationships on objects of the same type, the second step is not required. An important contribution of the extended topological map is that algorithms based on these four steps always access ESOs of the same spatial location.

```
Function Calc (R, S, Intersection)
/* we assume the R.k and S.l take their values in R */
begin
/*.....................................................................1st step */
for each r ∈ R do
    ΔV1 := MemberFaces(r.k, R.k);              /* decomposition into elementary faces */
    ΔV2:= ∅;
    /*.................................................................3rdstep */
    for each v1 ∈ ΔV1
        ΔV2:=ΔV2∪OwnerRegions(v1, S.l);        /* retrieve regions of S.l including face v1 */
    endfor
    /*.................................................................4th step */
    for each v2 ∈ ΔV2
        ΔV3 := MemberFaces(v2, S.l);           /* decomposition into elementary faces */
        RES = RES + (r, succ_val(v2, S.l), ΔV1 ∩ ΔV3);     /* build the result made */
                                               /* of tuple r and of the tuple owning region */
                                               /* v2 and of the intersection of r.k and v2 */
    endfor
endfor
end
```

Figure 10: Calculation operator involving the intersection function

5. Implementation

There are many ways to implement the GéoGraph graph. It could be implemented within a Network DBMS, an Object Oriented DBMS or as an extension of a Relational DBMS. We detail below a particular implementation based on the third approach, and on the connection of GéoGraph with the DBGraph model [Pucheral90] (as detailed in section 4.1). The key point of this implementation is a good data clustering. The objective is to partition the two graphs of figure 5 into separate segments which can be loaded separatly according to the needs of operations being executed.

In order to ease the data partitionning, tuples and values are stored separately (see figure 11). Since the domains form a partition of the set of values V, all the values varying over the same domain can be clustered in a separate segment. Taking advantage of vertical partitioning, the values of one domain can be loaded independently of the others. Similarly, since the relations form a partition of the set of tuples T, the tuples of one relation can be stored in one segment. Each object stored in a segment has a unique and invariant identifier (OID). Thus, tuples and values can be referenced by OID's.

In the definition of a DBGraph, an edge between a tuple and a value can be traversed in both directions. Consequently an edge is represented with two physical arcs: one from the tuple to the value, and also a reverse arc. A tuple is implemented as an array of OIDs, each referencing its attributes values, which are stored in separate domains. These OIDs materialize arcs from the tuples to the values. Reverse arcs are materialized by inverted lists attached to the values. Each inverted list is divided into a set of sublists so that there is one sublist per attribute varying on the domain of the value. All these sublists are referenced by an array attached to the corresponding value. Because of vertical partionning, it is possible to cluster all the inverted sublists of one attribute into one segment. Indices may be added on domain values to speed up selections on all attributes varying on the same domain.

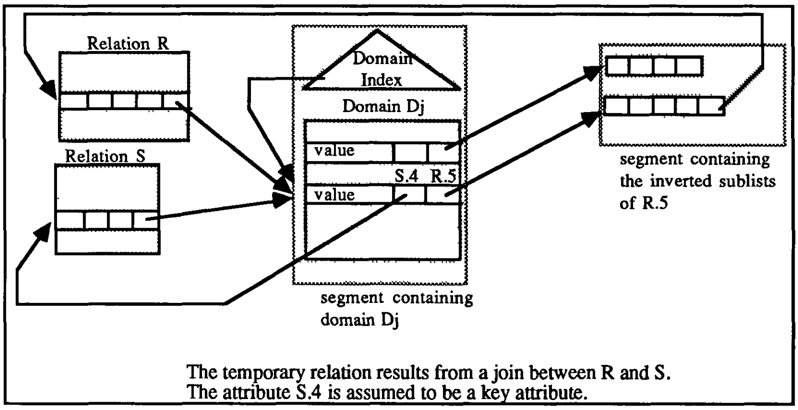

Relation R
Relation S
Domain Index
Domain Dj
value
S.4 R.5
value
segment containing domain Dj
segment containing the inverted sublists of R.5
The temporary relation results from a join between R and S.
The attribute S.4 is assumed to be a key attribute.

Figure 11: implementation of the DBGraph part

Consider now the implementation of the GéoGraph part. There is a direct mapping between a value of the spatial domain point and a node. Thus attributes of type point directly reference node values stored in a node domain. Values of the spatial domain region (resp. line) are stored as set of OIDs referencing face (resp. blade) values stored in a face (resp. blade) domain. The sets of OIDs materialize aggregation links from spatial objects to ESOs. Reverse links from blades and faces to the tuples, materilize reverse aggregation links from ESOs to tuples containing spatial objects. Face and blade domains are stored like DBGraph domains with inverted lists in order to materialize these links. It is not necessary to maintain inverted lists for region and line values. For some attributes it may be inefficient to store the values separatly from the tuples because they are accessed each time the tuple is accessed, and graph traversals of costly operations don't use links between the corresponding attribute values and the tuples. Values of these attributes can be stored directly in the tuples. For example, region and line attributes fall in this category. Spatial indices are maintained for the three domains node, blade and face. Values of these domains are clustured with the spatial indices which reinforce the contribution of the vertical partitionning, since algorithms based on extended topological maps favors access to ESOs of the same spatial location (see section 4.4).

Values ot the three domains face, blade and node have a complex structure that stores the topology of the ESO map (see figure 12). The main part of this topology is supported by the blade values. A blade value is represented by a record containing five fields: (i) the OID of the opposite blade, which materializes the α function; (ii) the OID of the next blade of the end-node, which materializes the σ function; (iii) the OID of the left face of the blade, which, used in conjunction with the α function, allows retrieval of the two faces bordering the blade; (iv) the OID of the end-node of the blade, which is necessary to access to the inverted list of the node; and (v) the OID of a coordinate list materializing the geometry of the blade. Coordinate lists are stored in a separate domain without inverted list. This domain is clustered with a geometrical index. A face value is the OID of any blade of the face. The value of one node contains only the OID of one blade reaching it. This information is sufficient for a node since the σ function gives the complete cycle of blades reaching it. Furthermore its geometry can be extracted from the geometry of one of its blade. In a similar way, the value of one face is composed of a record containing the OID of one blade of its boundary and a set of OID corresponding to the set of holes contained in this face. Applying a succession of α and σ functions to the blade referenced by the face gives the complete cycle of blades of its boundary. Its geometry can be obtained from the geometry of all of these blades.

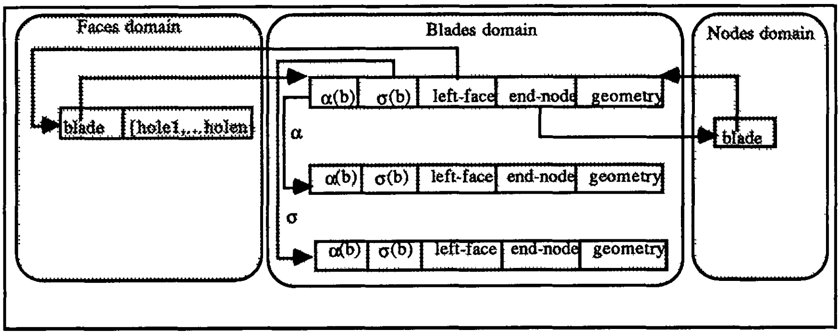


Figure 12: Spatial values representation

5. Summary and futur work

In this paper, we have presented the GéoGraph storage model, a toolbox supporting low layers of GIS in an extensible way. The definition of this model was given independently of implementation detail, in terms of a graph structure and primitive operations on that structure. This facilitates the description of the toolbox functionality, and supports our argument of general utility. Topological information and geometric information of several maps are incorporated in a single graph that directly supports geometric operations based on adjacency and containment relationships. This graph is based on the topological map theory guaranteeing that all updates on topological information are coherent, and providing a minimal set of operations to navigate through the graph.

Although GéoGraph is intended for various higher level data models, we illustrated the utilization of this storage model in the context of an extensible relational DBMS. In this context, the resulting GIS is itself extensible and can exploit fully the toolbox aspect of GéoGraph. We showed that the GéoGraph graph can be integrated with relational data in a straightforward fashion. Algorithms of the main geographical operations have been given in an abstract form using the basic primitives of GéoGraph. These algorithms, based on graph traversals, are simple and exibit desirable locality properties.

A specific implementation of the GéoGraph model has been proposed. This implementation avoids data duplication and shows that the GéoGraph graph can be partitioned to minimize disk trafic. Spatial data are clustered with spatial indices. This, combined with the space locality properties of the algorithms, reinforce the contribution of vertical partitionning. This implementation is currently being experimented in the framework of the GéoTropics system, an extensible GIS based on extensions of SQL [Bennis90].

Additional research will be useful in enhancing the GéoGraph model. For example, the decision to always overlap geographical maps has some drawbacks: operations involving only one map can be slowed down, since the number of elementary storage elements can be unnecessarily large. It may prove more efficient to selectively overlap layers based on the frequency of their joint use in queries[David90].

References

[Ansaldi85] Ansaldi S., De Floriani L., Falcidieno B., "Geometric Modeling of Solid Objects by Using Face Adjacency Graph Representation", ACM SIGGRAPH'85 Conf., San Francisco, USA, 1985.

[Bennis90] Bennis K., David B., Quilio I., Viémont Y., "GéoTROPICS: Database Support Alternatives for Geographic Applications", 4th Int. Symposium on Spatial Data Handling, Zurich, Switzerland, July 1990.

[Cori81] Cori R. & Vauquelin B., "Planar maps are well labeled trees", Can. J. Math., Vol. XXXIII, 1981.

[David89] David B., "External Specifications for the Cartographic DBMS", ESPRIT-TR 2427-0022 TROPICS Project, June 1989.

[David90] David B., Viémont Y., "Data Structure Alternatives for Very Large Spatial Databases", Sorsa colloquium 90, Fribourg, Deutshland, July 1990.

[Dufourd88] Dufourd J.F., "Algebraïc Specification and Implementation of the Topological Combinatorial Maps", PIXIM'88 proc., Paris, France, 1988.

[Dufourd89] Dufourd J.F, Gross C. et Spehner J.C., "A Digitizing Algorithm for the Entry of Planar Maps", Computer Graphics International'89, Leeds, Spring-Verlag, June 1989.

[Edmonds60] Edmonds J., "A Combinatorial Representation of Polyhedral Surfaces", Notices Amer. Math. soc. n°7, 1960.

[Herring87] Herring J.R., "TIGRIS: Topologically integrated geographic information system", Auto-Carto'8 proc., Baltimore, Maryland, USA, March 1987.

[Herring90] Herring J., "The definition and development of a Topologically Spatial

Data System", Photogrametry and Land Information Systems, Lausanne, Suisse, March 1989 (Published in 1990).

[Lienhardt89] Lienhardt P., "Subdivision of N-Dimensional Spaces and N-Dimensional Generalized Maps", 5th ACM Symposium on Computational Geometry, Sarbrüken, RFA, June 1989.

[Kinnea87] Kinnea C., "The TIGER Structure", Auto-Carto'8 proc., Baltimore, USA, March 1987.

[Meier82] Meier A., "A Graph Grammar Approach to Geographic Databases", Proc. of 2nd Int. Work. on Graph Grammars and their Application of Computer Science, october 1982, Lecture Notes in Computer Science, Springer Verlay, 1982.

[Meixler82] Meixler D., Sadlfeld A., "Storing, Retrieving and Maintaining Informations On Geographic Structures", Auto-Carto'7 Proc., Wachington, USA, March 1985.

[Morehouse85] Morehouse S., "A Geo-Relational Model for Spatial Informations", Auto-Carto'7 Proc. , Washington D.C., USA, 1985.

[Peuquet84] Peuquet Donna J., " A Conceptual Framework and Comparison of Spatial Data Models", Cartographica vol21 n°4, 1984.

[Pucheral90] Pucheral P., Thévenin J.M., Valduriez P., "Efficient main memory Data Management Using the DBGRAPH Storage Model", VLDB90 proc., Brisbane, Australia, August 1990.

[Sack87] Sack-davis R., Mcdonell K.J., "GEOQL - A Query Language for Geographic Information Systems", Australian and New-Zeland Association for the Advancement of Science Congress Townsville, Australie, August 1987.

[Samet85] Samet H., Webber E., "Storing a collection of Polygons Using Quadtree", ACM Transaction on Computer Graphics 3 (4), July 1985.

[Schaller87] Schaller J., 3The Geographical Information System (GIS) ARC/INFO", EuroCarto VI proceedings, Czechoslovakia, April 1987.

[Spooner90] Spooner R. "Advantages and Problems in the Creation and Use of Topologically Structured Database", Photogrametry and Land Information Systems, Lausanne, Suisse, March 1989 (Published in 1990).

[Stonebraker83] Stonebraker M., Rubenstein B., Guttman A., "Application of Abstract Data Types and Abstract Indices to CAD Databases", ACM Sigmod, San-Jose, 1983.

[Waugh87] Waugh T.C., Healey R.G., "The GEOVIEW Design, a Relational Approach to Geographical Data Handling", Int. J. Geographical Information Systems", 1(2), 1987.

[White79] White M. "A Survey of the Mathematics of Maps", Auto-Carto'4 proc., 1979.

TOPOLOGICAL MODELS FOR 3D SPATIAL INFORMATION SYSTEMS

Simon Pigot,
Environmental Systems Research Institute,[1]
380 New York St.,
Redlands, Ca. 92373
email: uucp: uunet!esri!atlas!simon
internet: spigot@esri.com

Abstract

The need for complex modelling and analysis of 3-dimensional data within a spatial information system (SIS) has been established in many fields. While much of the data that is currently being modelled seems to require "soft-edge" data structures such as grids or rasters, the need for certain types of complex topological modelling and analysis is clear. Current plane topology models such as the winged edge, widely used in computer aided design (CAD), are limited in the types of analysis that can be performed but useful because of their basis in the field of algebraic topology. This paper firstly reviews the neighborhood structure provided by current plane topological models. It then describes the derivation of a fundamental set of binary topological relationships between simple spatial primitives of like topological dimension in 3-space. It is intended that these relationships provide both a measure of modelling sufficiency and analytical ability in a spatial information system based on three dimensional neighborhoods.

1. Introduction

Modelling and analysis of 3-dimensional spatial phenomena has become a critical need in many applications, particularly the earth sciences. One of the traditional approaches to the modelling problem is to subset the sampled data from the 3D phenomena into individual spatial objects based upon theme or convenience; each spatial object can then be decomposed into a set of abstract geometric primitives - points, lines, faces and volumes; and a set of spatial relationships describing how the object may be reconstructed from these primitives. Analysis of the spatial phenomena requires not only the spatial relationships between the primitives required to reconstruct individual spatial objects, but also those relationships describing how the individual spatial objects interact. Such an approach is one method by which spatial objects may be modelled and analyzed according to theme or view in a larger model of the real phenomena.

[1]From April 11th, 1991, author's address will be: Centre for Spatial Information Studies, University of Tasmania, GPO Box 252C, Hobart, Tasmania, Australia, 7001. Internet email address: pigot@sol.surv.utas.oz.au

Topology is useful in both modelling and analysis because it provides simple and very useful spatial relationships, such as adjacency and connectivity. Topology can be thought of as the most primitive layer in a hierarchy of spatial relationships, where the next level of refinement is provided by the addition of familiar concepts based on a metric (ie. distance, direction etc.). Recent work by (Greasley 1988),(Kainz 1989) and (Kainz, 1990) in lattice theory seems to suggest that order relationships may exist at a similar level to topology.

Current topological models are either loosely or strongly based on a structure from algebraic topology known as the cell complex. The cell complex (in conjunction with graph theory) provides rules to govern the decomposition of a continuous 3D object into a finite number of points (0-cells), lines (1-cells), faces (2-cells) and volumes (3-cells). In governing the decomposition, the cell complex allows for the explicit description of three fundamental topological concepts: adjacency, connectivity and containment. Other relationships between individual objects such as whether two objects are disjoint or apart, may be provided by embedding individual cell complex(es) within a single cell or world cell and using the explicit relationships in combination to derive the particular relationship required. For example, it may be possible to analyze the explicit relationships to determine if two faces meet at a point (compare node connectivity of surrounding lines) or share a line (directly from adjacency). However, some relationships cannot be derived from these explicit relationships and may violate some of the rules of the cell complex. e.g. in 2-space (R^2) overlapping polygons (Egenhofer et. al. 1989); in 3-space (R^3), intersecting volumes or a face meeting another face at a point are all known to violate the rules of the cell complex governing the decomposition. In (Molenaar 1990) it is suggested that other relationships such as a line internal to a volume, also do not fit easily within the cell complex. From other work in 3D SIS (Youngmann 1988) and CAD (Weiler 1986), it appears that at least some of the modelling and analysis problems could be solved by combining the solid, surface and wire frame modelling approaches of CAD.

In this paper, the limitations of the cell complex are described by analyzing the direct and indirect topological relationships between cells that it provides. A layered set of fundamental binary topological relationships between simple lines, faces and volumes in R^3 based on point-set topology and extended from the work of (Egenhofer et. al. 1990) and (Pullar et. al. 1988) will be derived and presented. This paper and future research will attempt to integrate these intuitive yet powerful topological relationships and concepts with cell complex theory from algebraic topology since the power of the cell complex lies not in the nature and type of topological relationships that it allows, but in the ability to pose and solve topological problems as algebraic problems. It is expected that this approach will yield advantages both in modelling and analysis. For modelling purposes, the new topological relationships are intended to be used to ascertain the sufficiency of a cell complex based on 3D neighborhoods and provide insight into other useful structures such as lattices. Compactness and efficiency could be maintained by modelling only the coarsest topological relationships. For analysis purposes, a detailed set of basic topological relationships should provide either direct answers or at least the starting point of an answer, to complex spatial questions about the objects being modelled. In addition, enhancements to the fundamental modelling capability based on a complete set of topological relationships should allow boolean operations to be closed, i.e boolean operations may occur without the problem of not being able to model the result.

Sections 2 and 3 of this paper are concerned primarily with the cell complex. Section 2 introduces the necessary theory from algebraic topology and section 3 describes the current application of cell complex theory and the topological relationships which can be modelled. Section 4 introduces the necessary theory from point-set topology and presents the derivation of the new and richer set of topological relationships for R^3. Finally, section 5 concludes this paper with a summary of the results and the directions that will be taken in future research.

2. Topology

Topology is defined as the set of properties which are invariant under homeomorphisms (Alexandroff 1961) - one-to-one, continuous and onto transformations. Intuitively, it is easier to think of a homeomorphism as a kind of elastic transformation which twists, stretches and otherwise deforms without cutting. From the definition of topology as the study of those properties which remain invariant under homeomorphism, two objects are topologically equivalent if either can be transformed into the other using this type of elastic transformation. Clearly, metric properties such as distance, angle and direction are affected by homeomorphism and hence are not topological properties. It is the notion of homeomorphism which provides a fundamental or primitive set of spatial relationships (Chrisman 1987).

About Neighborhoods

The neighborhood of a point is any open set (ie. a set that does not include its boundary) that contains the point. Neighborhoods can be defined in any abstract manner, but the most common are those that have a metric interpretation. For example in 2D, the neighborhood of a point can be considered as any 2D "flat" disk containing that point.

About Manifolds

A manifold is an n-dimensional surface of which every point has a neighborhood topologically equivalent to an n-dimensional disk.This property is usually defined as local flatness. Manifolds are of interest because of their useful topological properties (in particular, the notion of orientation) which are inherited by the cells of a cell complex.

About Simplexes, Cells and Complexes

An n-simplex is the n-dimensional simplest geometric figure eg. a 1-simplex is a line, a 2-simplex a triangle and a 3-simplex a tetrahedron - in essence, an n-simplex has n+1 vertices and may be viewed as the smallest closed convex set containing the given vertices (Alexandroff 1961). An n-simplex is the homeomorph of an n-cell. eg. any closed polygon which does not have an internal boundary (ie. genus 0) is homeomorphic to a triangle or 2-simplex. Because of this topological equivalence all results for simplexes generalize to cells.

An n-simplex is a composite of n-1,n-2,...,1 simplexes. eg. a 2-simplex or triangle, is bounded by three 1-simplexes, which meet at three 0-simplexes. In (Egenhofer et. al. 1989) this property is termed "Completeness of inclusion".

An n-simplicial complex or more generally an n-cell complex is the homeomorph of an n-dimensional polyhedron whose faces are all (n-1)-cells, no two of which intersect except at a cell of lower dimension. In (Egenhofer et. al. 1989) this intersection restriction is termed "Completeness of incidence". With this restriction, an n-cell complex may inherit the properties of an n-manifold, thus accessing the topological properties of manifolds, the most important of which is orientation. The notion of orientation is usually applied to the 1-simplex by defining one of the bounding 0-simplexes or points as a point of origin and the other as a point of termination. Relative orientations can then be assigned to all higher simplexes according to the traversal of bounding 1-simplexes.

About Duality

Two dual operators which arise from these completeness axioms are termed boundary and coboundary, originally attributed to Poincaré (Corbett 1985). The boundary of an n-simplex is the incident set of n-1 simplexes. For example, a 3-simplex (tetrahedron) has 4 incident 2-simplexes, 6 incident 1-simplexes and 4 incident 0-simplexes. The coboundary of an n-simplex is the set of n+1-simplexes incident to the given n-simplex. For example, a 1-simplex may have two 2-simplexes cobounding it (one either side). The following table shows each cell and its dual, for R^3;

Primal	Dual
0-cell	3-cell
1-cell	2-cell
2-cell	1-cell
3-cell	0-cell

An important and powerful implication of duality is the fact that a primal may be represented and manipulated algebraically using its dual state. For example, 3-cells or volumes in a 3D SIS can be manipulated and represented by their dual state, the 0-cell or point.

3. Current Topological Models

Current topological models used either explicitly or implicitly in SIS and CAD fields are rather similar, despite the fact that CAD models are more generally used for 3D modelling and SIS are predominantly concerned with 2D models.

In SIS, 2D phenomena are assumed to be a connected set of points and lines (or a graph) which can be embedded in a 2-manifold - thus creating a set of connected and unconnected (or internal areas) (Corbett 1975),(Corbett 1979),(White 1983) and (White 1984). The application of the dual concepts of boundary and coboundary as described in the last section, provides connectivity and adjacency. Internal areas are described by simple application of homology theory. The data structures employed in such models are abstracted from graph theoretic concepts. Examples of systems built around these principles include DIME (Corbett 1975), ARC/INFO, TIGRIS (Herring, 1987), TIGER (Boudriault 1979).

In CAD and SIS surface modelling, even though 3D phenomena are being modelled, the current assumptions and resulting models are the same. The planar face of a 3D polyhedron is embedded in a 2-manifold and the embedded faces exist in 3D space resulting in a set of connected and unconnected (or internal) faces and volumes. The same application of the dual concepts of boundary and coboundary provides connectivity and adjacency e.g. (Corbett 1985) is a 3D extension of (Corbett 1975) and (Corbett 1979). Internal faces and volumes can be described by application of homology theory similar to that used for 2D SIS. The data structures employed in such models, such as the winged-edge model of (Baumgart 1975) and its later variants, e.g. (Braid et. al. 1978), (Woo 1985) and (Weiler 1985) are also based on graph theory and have been used extensively in CAD.

Both of these topological models can be described as vector, edge or boundary data structures and the particular topological relationships which are modelled can be classified using a system of relationships between 0,1 and 2D primitives specified in (Baer et. al. 1979) - see figure 1. Analysis of figure 1 shows that the main topological models in use, the winged-edge model in (Baumgart 1975) and the 2D map model in (Corbett 1975) and (Corbett 1979), both model the same set of relationships - EV and EF (from EV can derive VE, VV and EE, from EF can derive FE and FF, and from EF and EV together can derive VF and FV). Note that EV and EF give connectivity and adjacency corresponding with the boundary/coboundary principles of the cell complex - both models are basic applications of the cell complex. Most practical models do allow useful extensions that would normally be excluded by pure cell complex theory. For example, "dangling" lines - lines which are not connected at one or both ends to any other line.

The algebraic structure provided by cell complex theory has been reinvestigated in (Egenhofer et. al. 1989) and (Frank and Kuhn 1986) using concepts specified in (Giblin 1977) and (Moise 1977). This work has been applied to geological layers in (Carlson 1986) and to algorithms for editing triangular irregular networks in (Jackson 1989). The stated approach to the construction and maintenance of the cell complex is different to that taken previously because the construction and maintenance operations on the complex use topological concepts only - distance and other metric notions are not required.

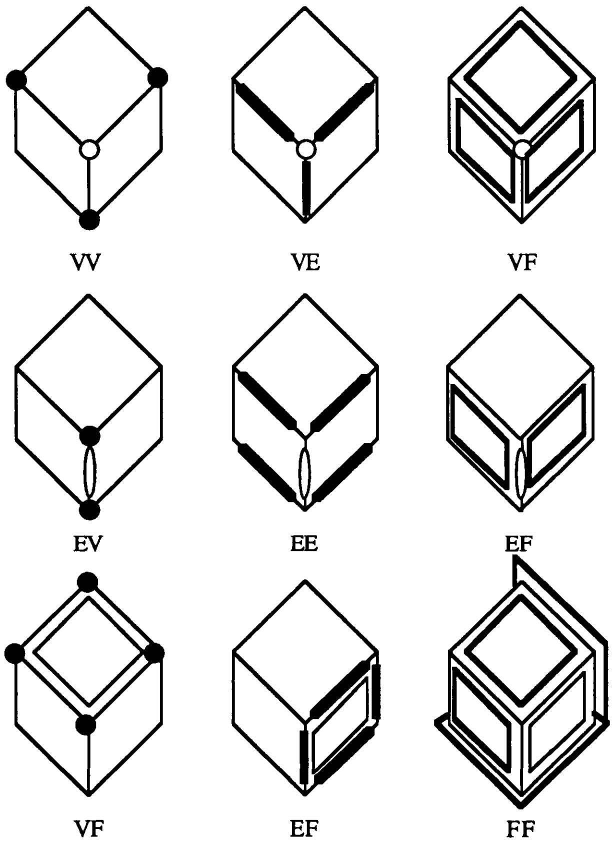


Figure 1 - 9 Relationships of (Baer, Henrion & Eastman 1979)

The intention is to avoid or at least minimize any inconsistency between the metric geometry and the topology that may be introduced by the limited precision arithmetic of computing devices (Franklin 1984). The other interesting aspect of (Egenhofer et. al.

1989) is that the construction and maintenance techniques are dimension independent.

In all models the necessary topological descriptions of faces with internal faces and volumes with internal volumes are described by the application of another branch of algebraic topology known as homology theory. e.g. (Corbett 1975), (Corbett 1979), (White 1983) and (Weiler 1985). Homology provides methods by which these internal faces and volumes may be detected by analysis of bounding cycles in cell complexes. In a wider sense homology groups give an indication of the connectivity present - internal faces and volumes may be regarded as homology group generators. In (Saalfeld 1989) other homology groups and additional homology theory are described and used in an attempt to determine the number of polygons resulting from the overlay of two maps.

4. Topological Relationships

What topological relationships may exist between abstract geometric primitives in euclidean 3-space? To answer a detailed question about the nature and type of all topological relationships is an attempt to classify the types and situations of manifolds. This is possible for R^1 (1-space) and R^2 (2-space) however, R^3 (3-space) has a number of quite difficult and unexpected situations which make general classification very difficult. See (Zeeman 1961) and (Alexandroff 1961). Fortunately, it is not necessary to attempt this. A number of assumptions about the nature of the relationships and the geometry of the n-cells involved can be made without limiting the power and application of the derived relationships. Specifically, only binary topological relationships between closed, connected (genus 0 - no internal holes) n-simplexes will be considered. The use of simplexes rather than cells is intuitive; simplicial complex theory is the starting point for the more generalized and advanced cell complex theory. Cells can be decomposed into simplexes in what is termed a simplicial decomposition, thus the results derived using simplicial complex theory can be generalized to cell complex theory via the decomposition.

In section 3, it was shown that cell complex theory as it is currently implemented in plane topology models allows a number of useful topological relationships such as adjacency and connectivity. In effect, cell complex theory allows n-dimensional adjacency (= connectivity in R^1), containment and the complement relationship of disjoint existing where no adjacency can be found. In essence, the main function of the cell complex is to allow specification of topological problems using algebraic methods, the definition of the algebraic operations being confined by the intersection rules (the set of allowable topological problems).

Point-set topology (classical topology) provides a much more intuitive view of topological relationships. In this paper, point-set binary topological relationships between 1-simplexes in R^3, 2-simplexes in R^3 and 3-simplexes in R^3 are based on consideration of the fundamental <u>boundary</u>, <u>interior</u> and <u>exterior</u> point-sets of any n-simplex in R^n. Additional point-sets are formed generically by embedding the n-simplex and its fundamental point-sets for R^n, within R^{n+1}. Consideration of the possible intersections of these point-sets with the boundary point-set of a second n-simplex then gives the fundamental topological relationships. The relationships are point-set topological relationships because they are derived from the intersection of these fundamental point-sets only.

The resulting binary topological relationships are very detailed. A number of methods can be chosen to aggregate or subdivide them into a hierarchy of detail. The method of aggregation chosen in this paper is consistent with the topological notion of homeomorphism. Each of the resulting binary topological relationships is considered to be the union of the two n-simplex point sets involved. Some topological relationships are then homeomorphic and can be replaced by a single homeomorph. The resulting tree structure then provides two levels of detail, the most descriptive relationships being found at the "leaves" of the tree. Further subdivision and grouping could also occur by considering the dimension of the spatial intersection between the two n-simplexes in each relationship as proposed in (Egenhofer et. al. 1990).

In all of the following discussion, a 1-simplex is called an interval, a 2-simplex is called a face and a 3-simplex is called a volume.

Theoretical Background

All results used and derived in this section are for metric topological spaces since metric topological spaces are most commonly used for modelling purposes. Metric topological spaces are a subset of general topological spaces.

An n-simplex in R^n divides R^n into three useful and intuitive point-sets, well known in point-set topology; eg. (Kasriel 1971)

Interior $^\circ$ set of an n-simplex **C**: a point x is an interior point of **C** provided there exists an open subset **U** such that *x* is an element of **U** and **U** is strictly contained within **C**. The union of all such points is the interior set.

Boundary set ∂ of an n-simplex **C**: $\mathbf{C} - {}^\circ$

Exterior set of an n-simplex **C**: Complement of **C**.

A simple and complete method can be found for finding all topological relationships between two closed, connected n-simplexes. In (Pullar et. al. 1988), (Driessen 1989) and (Egenhofer et. al. 1990) only the intersection of the boundary and the interior point-sets of the two n-simplexes is used to derive topological relationships. In this paper, a more powerful and fundamental method is used which is based on the set intersection of the boundary, interior and exterior point-sets of an n-simplex n_1 and the boundary, interior and exterior sets of another n-simplex n_2 in R^n. In practice, the derivation of relationships can be simplified by considering the possible set intersection of the boundary point-set of n_1 and the interior, exterior and boundary point-sets of n_2, since the boundary set of n_1 naturally defines the interior and exterior point-sets of n_1 and governs their possible relationships with the sets of n_2. Further detail can then be added to each relationship if required by considering the set intersection of the interior and exterior sets of n_1 with those of n_2.

Up till now the definitions of the fundamental point-sets of an n-simplex have been given in terms of an n-simplex in R^n, however in order to analyze intersections between n-simplexes in R^{n+1} it is necessary to consider what happens to the simplex and its point-sets in R^n when they are embedded in R^{n+1}. This is of particular importance to this research, since the aim is to derive topological relationships between 1-simplexes, 2-simplexes and 3-simplexes in R^3.

The closed boundary and open interior and exterior point-sets of an n-simplex in R^n are all closed point-sets when considered relative to R^{n+1} since the union of these point-sets is an n-manifold equivalent to R^n, and R^n itself is a closed point-set in R^{n+1}. Since the intersection process is reliant upon the existence of these three point-sets then we have a problem, the solution to which can be found by considering the dimension of the n-manifold created from the union of these point-sets and the dimension of the space in which they to be embedded. In R^n, we are considering the intersection of the boundary, interior and exterior point-sets of two n-simplexes in the same n-manifold which is equivalent to R^n. In R^{n+1}, we consider not only the situation in R^n where both n-simplexes are in the same n-manifold, but also the complement situation which occurs when both n-simplexes are in different n-manifolds. Clearly any intersection between the boundary, interior and exterior point-sets of the two n-simplexes will always occur where the two n-manifolds meet, hence if the open/closed point-set properties of the interior, exterior and boundary point-sets of an n-simplex are considered strictly relative to the n-manifold formed by their union, then their open/closed point-set properties are preserved and can be used without loss of generality regardless of the dimension of the space in which the n-manifold(s) created from their union are embedded.

It is now necessary to find a simple and comprehensive way of analyzing the intersection possibilities between two n-simplexes in R^{n+1} excluding the subset formed specifically for R^n when both n-simplexes are in the same n-manifold. This can be done by choosing a specific embedding of such an n-manifold or equivalently R^n, in R^{n+1}. If R^n and R^{n+1} are metric spaces with standard orthogonal basis vectors (or coordinate system axes) then if we choose the embedding such that the n orthogonal basis vectors of R^n are coincident with n of the n+1 orthogonal basis vectors of R^{n+1}, then R^n disconnects R^{n+1} into two open point-sets corresponding to the opposing directions of the n+1th orthogonal basis vector of R^{n+1}. Using this fact the derivation method for possible intersections between n-simplexes in R^{n+1} can be extended simply by considering those intersection combinations involving either or both of the two new point-sets resulting from the embedding.

In the summary of the theory and the rest of this paper, the generic term set is used in place of point-set. The theory can now be summarised in five steps as follows;

1. Formulate the boundary, interior and exterior sets of an n-simplex n_1 in R^n.
2. Derive basic relationships based on all possible set intersections of the boundary set of a second n-simplex n_2 and the interior, boundary and exterior sets of the n-simplex n_1 from step 1.
3. Consider the union of the interior, exterior and boundary sets of any n-simplex in R^n as an n-manifold equivalent to R^n with the definition of the open/closed properties of these sets strictly relative to R^n.
4. Disconnect R^{n+1} into two new open sets by choosing an embedding of R^n (created in step 3) in R^{n+1} such that the n orthogonal basis vectors of R^n are coincident with n of the n+1 orthogonal basis vectors of R^{n+1}.
5. Derive additional relationships based on the possible set intersections of the boundary set of an n-simplex n_2 with the boundary, interior and exterior sets of the a second n-simplex n_1 with the boundary set of n_2 intersecting either or both of the two new sets predicted in step 4.

<u>Intervals (1-simplexes)</u>

The boundary, interior and exterior sets of an interval i_1 in R^1 are shown in figure 2.

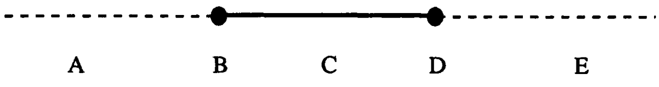


Figure 2 - The exterior, boundary and interior sets of an interval in R1

Note that there are two distinct closed boundary sets (B and D), two distinct open exterior sets (A and E) and a single open interior set (C). The union of the sets A,B,C,D and E is a 1-manifold equivalent to R^1. All possible binary topological relationships between two intervals in R^1 can then be derived by choosing any two points x and y forming the boundary set of a second interval i_2, either from the same set or each from a different set, and making these the boundaries of an interval joining them. The created interval i_2 will then either intersect interval i_1 in some way or be disjoint from it. e.g. If both points x and y are chosen from set A (the left exterior set) then the created interval i_2 will not intersect i_1. The unique combinations and their spatial interpretations are shown in figure 3.

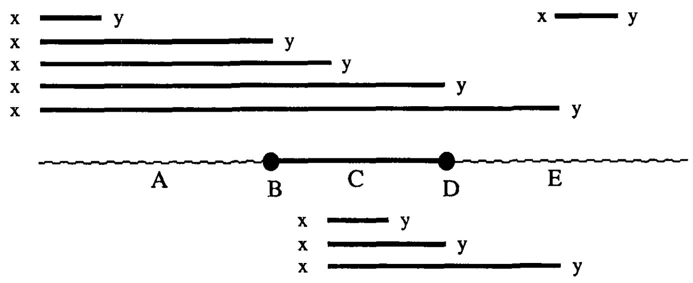


Figure 3 - Possible choice combinations of the two boundary points x,y from the boundary, interior and exterior sets of an interval in R1

From figure 3, it is possible to distinguish those choices which give distinct relationships and name these distinct relationships as follows; (ε = element of a set)

x ε boundary set B or D, y ε boundary set D or B	->	i_1 equals i_2
x,y ε exterior set A or E	->	i_1 and i_2 are disjoint
x,y ε interior set C	->	i_1 contains i_2
x ε exterior set A or E, y ε interior set C	->	i_1 and i_2 overlap
x ε exterior set A or E, y ε boundary set B or D	->	i_1 meets i_2
x ε boundary set B or D, y ε interior set C	->	i_1 and i_2 share common bounds

Note that these six relationships are the same as those derived in (Pullar et. al. 1988). The names given to the six distinct relationships are also taken from (Pullar et. al. 1988).

If we define the open/closed properties of these sets strictly relative to R^1 then these sets and the set relationships in R^1 are preserved when the five sets A,B,C,D and E whose union comprises R^1 are embedded in R^2. As for the new sets created by the embedding; if the embedding of R^1 in R^2 is chosen such that the basis vector of R^1 corresponds to one of the two orthogonal basis vectors of R^2, then R^2 will be divided into two open sets F and G, separated by a third set corresponding to R^1. The situation is shown in figure 4. R^1 is represented by the line L.

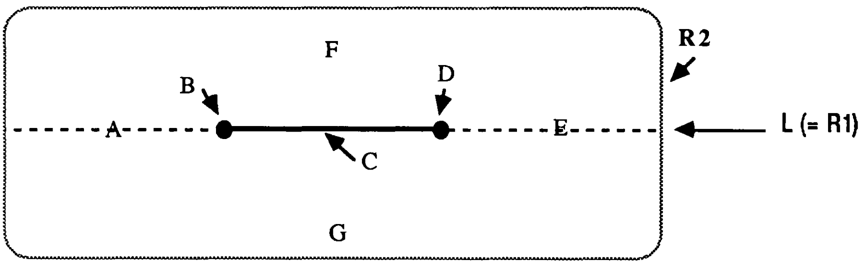


Figure 4 - New point sets F & G obtained by embedding R1 in R2

All possible binary topological relationships between intervals in R^2 can be derived in the same way as for R^1, by choosing two points either from the same set or from a different set and making these the boundary of an interval. Since those relationships derived in R^1 apply without modification in R^2, only the new combinations where x,y are elements of either or both sets F and G will be considered.

The set relationships can be divided into groups by examination of figure 4. The first group occurs when both boundary points are in the sets A,B,C,D or E which comprise R^1 (the line L) and has already been considered above. The second group occurs when either one or both of the boundary points of i_2 are contained within either F or G. The spatial situation corresponds to the interval i_2 being either left or right of the line L. The possible combinations and their spatial interpretations are shown in figure 5a.

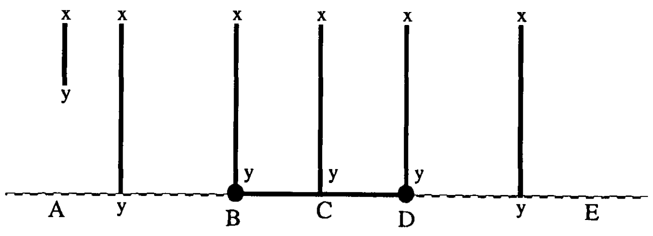


Figure 5a - Intersection between the boundary point y of interval i2 and the boundary, interior and exterior sets of interval i1 when boundary point x of i2 is always chosen from the point set F (or equivalently, G).

The following set relationships may be distinguished based upon which sets the boundary points x and y intersect;

$x \in$ set F, $y \in$ set F	OR		
$x \in$ set G, $y \in$ set F	OR		
$x \in$ set F or set G, $y \in$ exterior set A or E		->	i_1 and i_2 disjoint

x ε set F or set G, y ε boundary set B or D	->	i_1 meets i_2
x ε set F or set G, y ε interior set C	->	i_1 intersects i_2

The third group occurs when one of boundary points of i_2 is an element of F and the other is an element of G, indicating that the interior set of interval i_2 intersects the line L (the boundary, exterior and interior sets of i_1) at a point. The possible combinations and their spatial interpretations are shown in figure 5b.

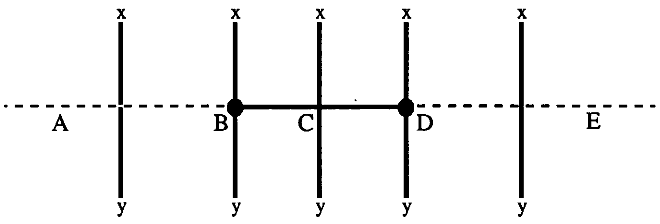


Figure 5b - Intersection between the interior set of interval i2 and the boundary, interior and exterior sets of interval i1 when x and y are chosen from the point sets F and G respectively.

The three resulting relationships are distinguished according to which set of i_1 that the interior of i_2 intersects (in fact, the three possible relationships between a single point and the interior, boundary and exterior sets of an interval);

x ε set F, y ε set G, intersect exterior set A or E	->	i_1 and i_2 disjoint
x ε set F, y ε set G, intersect boundary set B or D	->	i_1 intersects i_2
x ε set F, y ε set G, intersect interior set C	->	i_1 and i_2 cross

By consideration of both these groups, the only new relationships which result are intersect and cross, making a total of 8 relationships between intervals in R^2. For R^3 also, no new relationships result because embedding the scheme for R^2 shown in figure 4, in R^3 produces two new sets as a result. The same process of reduction for R^2 reveals no new relationships - hence there are eight relationships between intervals in R^3.

To reduce these 8 relationships in detail, the union of the boundary and interior points-sets of i_1 and i_2 is considered. Relationships can then be eliminated which are homeomorphic. For intervals, this results in relationships; meet, overlap, contains, equal, common-bounds all being homeomorphic to a single interval. Thus, the complete two layer hierarchy of binary topological relationships between intervals (1-simplexes) in R^3 is shown in figure 6.

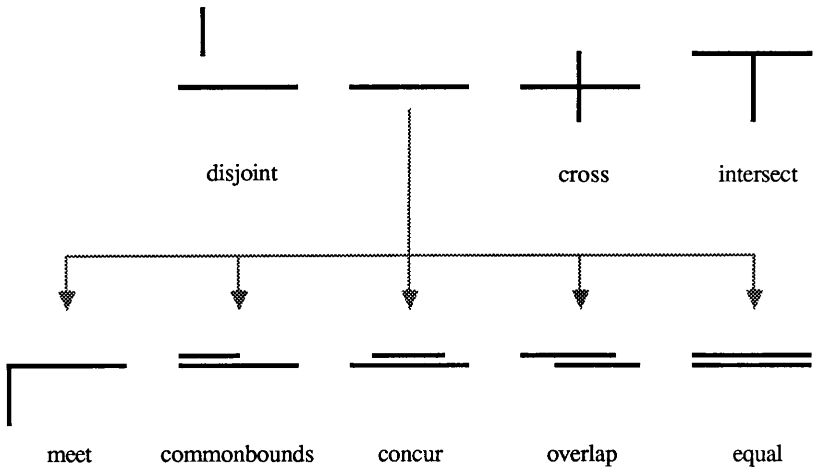


Figure 6 - The eight unique binary topological relationships between 1-cells in R3

Faces (2-simplexes)

The boundary, interior and exterior sets of a face (or 2-simplex) a_1 in R^2 are shown in figure 7.

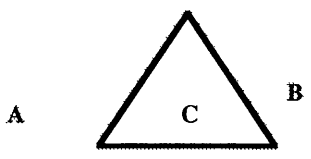


Figure 7 - Exterior (A), Boundary (B) and Interior (C) point-sets of a face in R2

Note that there is a single closed boundary set (B), a single open exterior set (A) and a single open interior set (C). The union of sets A,B and C is a 2-manifold equivalent to

R^2. All possible binary topological relationships between faces in R^2 can then be derived from the possible set relationships between the boundary, interior and exterior sets A, B and C of a_1 and the boundary set X of a_2. e.g. if the boundary set X of a_2 is contained within the interior set C, then the face a_2 will be contained within a_1. The combinations matrix showing the possible relationships between the boundary of the face a_2 and the exterior, boundary and interior sets A,B, and C of a_1 is shown in table 1.

Exterior A	x			x		x	x
Boundary B		x		x	x	x	
Interior C			x	x	x		x

Table 1: Set intersection relationships between the boundary set of a2 and the interior, exterior and boundary sets of a1 in R2

Note that the seventh relationship in the last column of table 1 is not possible in R^2 because of the restriction to closed, connected faces.

The six distinct relationships and their names are the same as those in (Egenhofer et. al. 1990). The spatial interpretations are shown in figure 8.

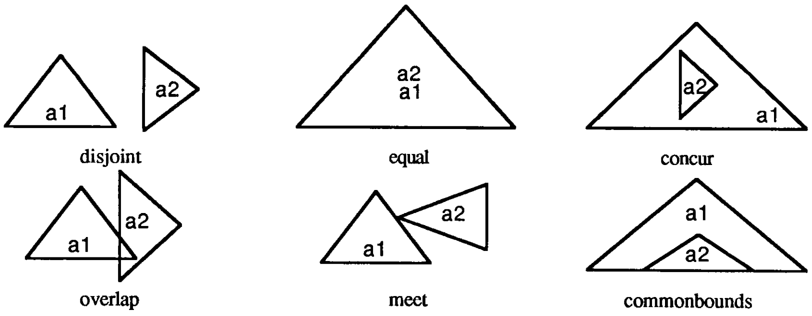


Figure 8 - Six possible relationships between faces based on the intersection of the boundary set of face a2 and the exterior, boundary and interior sets of face a1 in R2

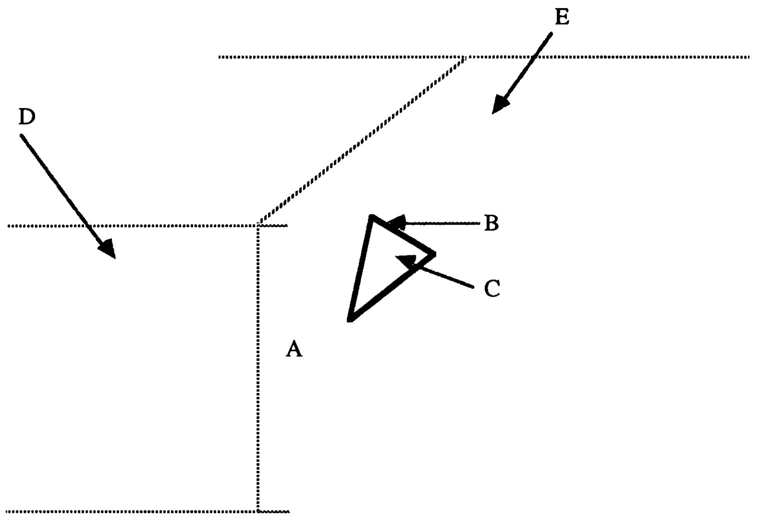


Figure 9 - New point sets D & E obtained by embedding the union of the boundary (B), exterior (A) and interior (C) of a face al in R3 (A U B U C = R2).

If we define the open/closed properties of these sets strictly relative to R^2 then these properties and the set relationships in R^2 are preserved when the 2-manifold (equivalent to R^2) formed by their union is embedded in R^3. If the embedding is chosen such that any two orthogonal basis vectors of R^2 are coincident to two of any three orthogonal basis vectors of R^3 then R^2 disconnects R^3 into two open sets with the third open set corresponding to R^2 itself. The situation is shown in figure 9.

All possible binary topological relationships between faces in R^3 can be derived in the same way as for R^2, by considering the possible set relationships between boundary set of a face a_2 and the boundary, interior and exterior sets of the face a_1 plus the two new sets D and E which result from embedding R^2 in R^3. Since all set relationships derived for R^2 are preserved in R^3, only the combinations involving the new sets D and E will be considered.

By examination of figure 9, the set relationships can be divided into two groups. The first group represents the situation where the boundary set X of a_2 is contained within the plane **P** formed from the union of the interior, exterior and boundary sets of a_1. This situation corresponds to faces in R^2 and was considered above. The second group corresponds to the situation where the boundary set X of a_2 intersects either D or E but not both. This corresponds to the spatial situation where a_2 is completely on one side of the plane **P** formed by the boundary, interior and exterior sets A,B and C of a_1. In this situation, the boundary set X of a_2 may intersect the plane **P** and hence the boundary, interior and exterior sets A,B and C or not at all. All combinations are shown in table 2.

	1	2	3	4	5	6	7	8
Exterior A				X		X	X	X
Boundary B			X		X	X		X
Interior C		X			X		X	X
Above D Below E	X	X	X	X	X	X	X	X

Table 2: Set intersections between the boundary set X of a2 and the interior (A), exterior (B) and boundary (C) sets of a1 in R3 when a1 intersects only one of the sets D or E.

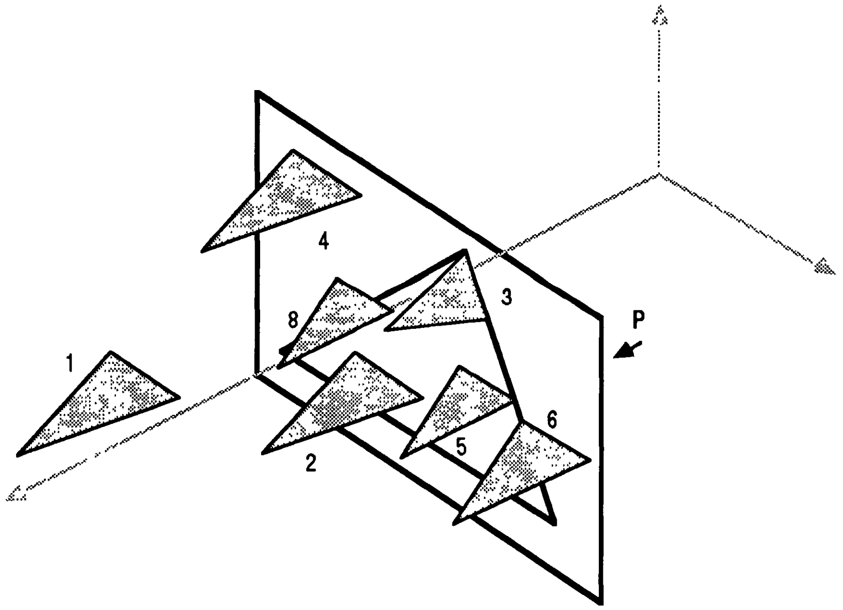


Figure 10 - Relationships formed by the intersection of the boundary set X of a face a2 with the boundary, interior and exterior sets (A,B and C) of a face a1 when the boundary set of of the face intersects the point-set D (or E). a2 is shown shaded, however only the black outline is the boundary set of a2

Since the topological relationships are the same no matter which set D or E on either side of the plane **P** the boundary set of a_2 intersects, the combinations are shown in the table with the marker offset between D and E. Note that relationship 7 is not possible between two closed connected simplexes. The other seven relationships are shown spatially in figure 10.

The third group of relationships occurs when the boundary set X of a_2 intersects both D and E and hence must intersect the sets A,B and C of a_1 at an interval whose boundaries correspond to two points from the boundary set X of a_2 and interior corresponds to the interior set Y of a_2. The possible combinations between the boundary set X of a_2 and the boundary, interior and exterior sets A,B and C of a_1 when the boundary set X intersects both D and E as well, are shown in table 3.

	9	10	11	12	13	14	15
Exterior A			X		X	X	X
Boundary B		X		X	X		X
Interior C	X			X		X	X
Above D	X	X	X	X	X	X	X
Below E	X	X	X	X	X	X	X

Table 3: Set Intersections between the boundary of set X of a2 and the exterior (A), boundary (B) and interior(C) sets of a1 in R3 when the boundary of a2 intersects both of the sets D and E.

The spatial interpretations are shown in figure 11. Note that for relationship 10 in column two, the interior set of the face a_2 may be used to derive a second possibility. These relationships are marked 10a and 10b in the spatial interpretations of these relationships, shown in figure 11. In addition, relationship 14 is not possible between closed, connected faces.

By examination of all relationships in figures 8, 10 and 11, the number of unique relationships between faces in R^3 is fourteen since relationships 1,4 and 11 are particular types of the disjoint relationship shown in figure 8 and relationships 3, 6 and 13 are particular types of the meet relationship shown in figure 8.

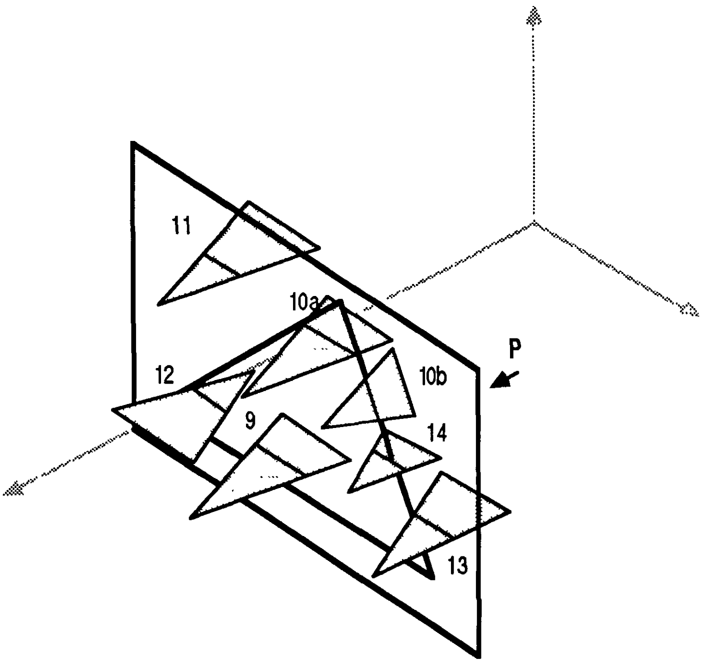


Figure 11 - Relationships formed by the intersection of the boundary set X of a face a2 with the boundary, interior and exterior sets (A,B and C) of a face a1 when the boundary set of of the face intersects both point-sets D and E (passes throught the plane P formed from the union of the boundary, interior and exterior sets of a1. Although a2 is shown shaded, only the black outline is the boundary set

To reduce these fourteen relationships in detail, the union of the boundary and interior points-sets of a_1 and a_2 in each relationship is considered. Relationships which are homeomorphic can then be reduced to their homeomorphs. Thus, the complete two layer hierarchy of binary topological relationships between faces (2-simplexes) in R^3 is shown in figure 12.

Volumes (3-simplexes)

The boundary, interior and exterior sets of a volume (or 3-simplex) v_1 in R^3 are the same as for a face in R^2 (Figure 7). There is a single closed boundary set (B), a single open exterior set (A) and a single open interior set (C) just as there was for faces in R^2 in the previous section. The union of sets A,B and C is a 3-manifold equivalent to R^3. All possible binary topological relationships between volumes in R^3 can then be derived from the possible set relationships between the boundary, interior and exterior

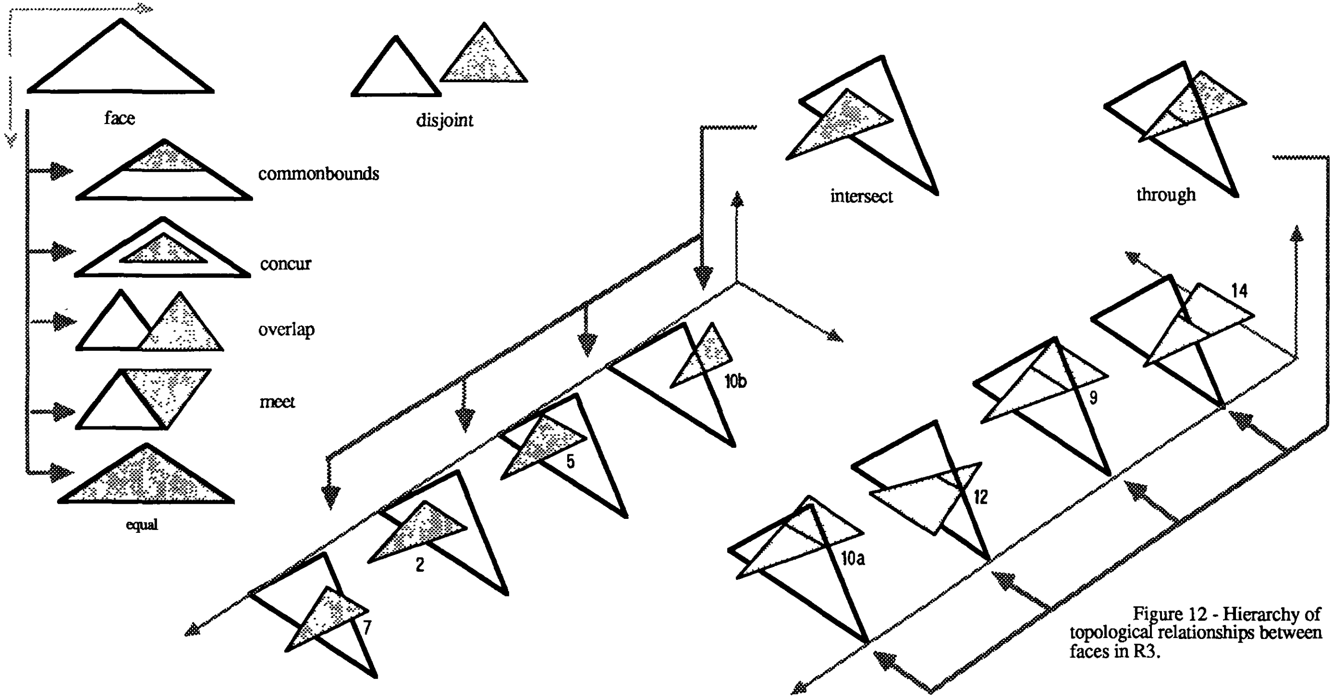


Figure 12 - Hierarchy of topological relationships between faces in R3.

sets A, B and C of v_1 and the boundary set X of v_2. e.g. if the boundary set X of v_2 is contained within the interior set C, then the volume v_2 will be contained within v_1. The combinations matrix showing the possible relationships between the boundary of a volume v_2 and the exterior, boundary and interior sets A,B, and C of v_1 is shown in table 4.

Exterior A	X			X		X	X
Boundary B		X		X	X	X	
Interior C			X	X	X		X

Table 4: Set intersection relationships between the boundary set of v2 and the interior, exterior and boundary sets of v1 in R3

Note that the seventh relationship in the last column of table 4 is not possible in R^3 because of the restriction to closed, connected volumes. Not surprisingly the relationships are the same as those between closed, faces in R^2.

The six distinct relationships and their names are the same as those used in (Egenhofer et. al. 1990). The spatial interpretations are shown in figure 13.

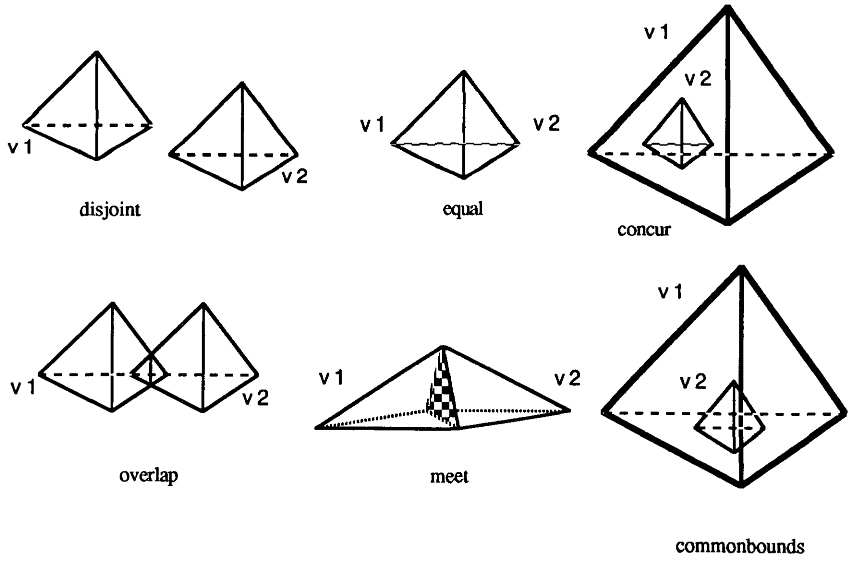


Figure 13 - Six fundamental relationships between the boundary set of a volume v2 and the boundary, exterior and interior sets of a volume v1

Note that is also possible to use the sixteen different boundary-interior set intersection combinations and the theory shown in (Egenhofer et. al. 1990), to derive the same eight relationships between volumes or 3-simplexes. The only change in the theory required is the use of an extension of the Jordan-Brouwer separation theorem to R^3, given in (Alexander 1924).

To reduce these 8 relationships in detail, the union of the boundary and interior sets of v_1 and v_2 in each relationship is considered. Relationships which are homeomorphic can then be eliminated. For volumes in R^3, this results in meet, overlap, contains, equal and common-bounds all homeomorphic to a single volume.

5. Conclusions and Future Research

The final aim of this research is a compact and powerful spatial information system for 3D modelling and analysis. Since topological situations in 3-space are complex and difficult, a natural starting place for the development and investigation of a 3D neighborhood topological model is to limit the types of relationships to those that may occur between simplexes since they may be generalized to complex problems via a simplicial decomposition. The topological relationships limiting the cell complex as currently used in 3D topological models for SIS and CAD have been described. To provide a better theoretical basis for 3D situations, a generic and reusable method for deriving fundamental point-set topological relationships between two closed, connected n-simplexes (genus zero) in R^{n+1} (and higher dimensions) has been developed. The generalized method of derivation can be summarised in two steps;

1. Consider the set intersection of the boundary set of a single n-simplex n_2 with the boundary, interior and exterior sets of a second n-simplex n_1 in R^n.
2. Extend these relationships by including either or both of the two additional sets created by embedding the n-manifold created from the union of the boundary, interior and exterior sets of n_1 in R^{n+1}.

Using this method, the derived sets of binary topological relationships for R^3 have been presented as a two-layer hierarchy. Relationships in the first layer are created by considering the union of the boundary and interior sets of the two n-simplexes and replacing those relationships in the second layer which are homeomorphic with a homeomorph. The results are as follows;

	Second Layer - Fundamental	First Layer - Aggregated
1-simplexes in R^3	8	4
2-simplexes in R^3	14	4

3-simplexes in R^3	6	2

It is interesting to note that the relationships between 3-simplexes verify the correctness of the extended set of relationships between faces or 2-simplexes in R^3. Each relationship between faces or 2-simplexes implied by the eight 3-simplex relationships is predicted within the extended set of 2-simplex face relationships. Similarly, the relationships between 2-simplexes verify the extended set of 1-simplex relationships in R^3.

Future research will concentrate on the development of a 3D neighborhood topological model for SIS, the basis for the modelling sufficiency and analytical power of this model will be the relationships derived in this paper. In addition, other hierarchies of these relationships based on set and order theory will be investigated.

6. References

Alexander, J.W., 1924, On the Subdivision of 3-Space by a Polyhedron, *Proceedings of the National Academy of Science*, vol. 10, pp. 6-12.

Alexandroff, P., 1961, *Elementary Concepts of Topology* (Dover Publications: USA).

Baer, A., C. Eastman, and M. Henrion, 1979, Geometric Modelling: A Survey, *Computer Aided Design*, vol. 11, no. 5, pp. 253-272.

Baumgart, B.G., 1975, A Polyhedron Representation for Computer Vision, *American Federation of Information Processing Societies (AFIPS Conference), Proceedings of the NCC*, vol. 44, pp. 589-596.

Boudriault, G., 1987, Topology in the TIGER file, *Proceedings of the Eighth International Symposium on Computer Assisted Cartography (AUTOCARTO 8)*, pp. 258-263.

Braid, I.C., R.C. Hillyard, and I.A. Stroud, 1978, Stepwise Construction of Polyhedra in Geometric Modelling, *Mathematical Methods In Computer Graphics and Design* (K.W. Brodlie ed.), pp. 123-141, Academic Press.

Carlson, E., 1986, Three Dimensional Conceptual Modelling of Subsurface Structures, *Technical Papers of the ACSM/ASPRS Annual Convention*, Baltimore, Maryland, USA, vol. 3, pp. 188-200.

Chrisman, N.R., 1987, Challenges for Research in Geographic Information Systems, *International Geographic Information Systems (IGIS) Symposium*, vol. 1, pp. I-101 to I-112.

Corbett, J.P., 1975, Topological Principles In Cartography, *Proceedings of the International Symposium on Computer Assisted Cartography (AUTOCARTO 2)*, pp. 61-65.

Corbett, J.P., 1979, Topological Principles in Cartography, Technical Report No. 48, US Bureau of Census, Washington, D.C.

Corbett, J.P., 1985, A General Topological Model For Spatial Reference, *Spatially Oriented Referencing Systems Association (SORSA) Workshop* (J.P. van Est ed.), Netherlands, pp. 9-24.

Driessen, R.J., 1989, A Model for Land Parcels in a LIS, Master of Surveying Thesis, School Of Surveying, University Of Tasmania.

Egenhofer, M.J., A.U. Frank and J.P. Jackson, 1989, A Topological Model for Spatial Databases, *Design and Implementation of Large Spatial Databases* (A. Buchmann, O. Gunther, T.R. Smith and Y.-F. Wang eds) SSD 89, vol. 409, pp. 271-286, Springer-Verlag.

Egenhofer, M.J. and J.R. Herring, 1990, A Mathematical Framework for the Definition of Topological Relationships, *Proceedings of the Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, vol. 2, pp. 803-813.

Frank, A.U. and W. Kuhn, 1986, Cell Graphs: A Provable Correct Method for the Storage of Geometry, *Proceedings of the 2nd International Conference on Spatial Data Handling*, Seattle, Washington, USA, pp. 411- 436.

Franklin, Wm. R., 1984, Cartographic Errors Symptomatic of Underlying Algebraic Problems, *Proceedings of the First International Symposium on Spatial Data Handling*, Zurich, Switzerland, vol. 1, pp. 190-208.

Giblin, P.J., 1977, *Graphs, Surfaces and Homology*, (Chapman and Hall: U.K.).

Greasley, I., 1988, Data Structures to Organize Spatial Subdivisions, Report 79, University of Maine at Orono, Dept. Surveying Engineering.

Herring, J.R., 1987, TIGRIS: Topologically Integrated Geographic Information System, *Proceedings of the Eighth International Symposium on Computer Assisted Cartography (AUTOCARTO 8)*, Baltimore, Maryland, USA, pp. 282-291.

Jackson, J.P., 1989, Algorithms for Triangular Irregular Networks Based on Simplicial Complex Theory, *Technical Papers of the ACSM/ASPRS Annual Convention*, Baltimore, Maryland, USA, vol. 4, pp. 131-136.

Kainz, W., 1989, Order, Topology and Metric in GIS, *Technical Papers of the ACSM/ASPRS Annual Convention*, Baltimore, Maryland, USA, vol. 4, pp. 154-160.

Kainz, W., 1990, Spatial Relationships - Topology Versus Order, *Proceedings of the Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, vol. 2, pp. 814-819.

Kasriel, R., 1971, *Undergraduate Topology*, (W.B. Saunders Company: USA).

Mendelson, B., 1968, *Introduction to Topology*, 2nd Edition, (Allyn & Bacon: USA).

Moise, Edwin E., 1977, *Geometric Topology In Dimension 2 and 3*, Springer-Verlag.

Molenaar, M., 1990, A Formal Data Structure For Three Dimensional Vector Maps, *Proceedings of the Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, vol. 2, pp. 830-843.

Pullar, D.V. and M.J. Egenhofer, 1988, Toward Formal Definitions of Topological Relations Among Spatial Objects, *Proceedings of the Third International Symposium on Spatial Data Handling*, Sydney, Australia, pp. 225-243.

Saalfeld, A., 1989, The Combinatorial Complexity of Polygon Overlay, *Proceedings of the 9th International Symposium on Computer Assisted Cartography (AUTOCARTO 9)*, Baltimore, Maryland, USA, pp. 278-288.

Spanier, E.H., 1966, *Algebraic Topology*, McGraw Hill Book Company.

Weiler, K., 1985, Edge-Based Data Structures for Solid Modelling Environments, *IEEE Computer Graphics and Applications*, vol. 5, No. 1, pp. 21-41, IEEE, USA.

Weiler, K., 1986, Topological Structures for Geometric Modelling, Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, New York, USA.

White, M., 1983, Tribulations of Automated Cartography and how Mathematics Helps, *Proceedings of the 6th International Symposium on Computer Assisted Cartography (AUTOCARTO 6)*, Canada, vol. 1, pp. 408-418.

White, M., 1984, Technical Requirements and Standards for a Multipurpose Geographic data system, *The American Cartographer*, vol. 11, No. 1, pp. 15-26.

Woo, T.C., 1985, A Combinatorial Analysis of Boundary Data Schema, *IEEE Computer Graphics and Applications*, vol. 5, No. 3, pp. 19-27.

Youngmann, C., 1988, Spatial Data Structures for Modelling Subsurface Features, *Proceedings of the Third International Symposium on Spatial Data Handling*, Sydney, Australia, pp. 337-341.

Zeeman, R., 1961, The Topology of the Brain and Visual Perception, *Topology of 3-manifolds and Related Topics* (ed. M.K. Fort), (Prentice-Hall: USA).

The Reactive-tree: A Storage Structure for a Seamless, Scaleless Geographic Database

Peter van Oosterom*

TNO Physics and Electronics Laboratory,
P.O. Box 96864, 2509 JG The Hague, The Netherlands.
Email: oosterom@fel.tno.nl

Abstract

This paper presents the first fully dynamic and reactive data structure. *Reactive data structures are vector based structures tailored to the efficient storage and retrieval of geometric objects at different levels of detail. Geometric selections can be interleaved by insertions of new objects and by deletions of existing objects. Detail levels are closely related to cartographic map generalization techniques. The proposed data structure supports the following generalization techniques: simplification, aggregation, symbolization, and selection. The core of the reactive data structure is the* Reactive-tree, *a geometric index structure, that also takes care of the selection-part of the generalization. Other aspects of the generalization process are supported by introducing associated structures, e.g. the* Binary Line Generalization-tree *for simplification. The proposed structure forms an important step in the direction of the development of a seamless, scaleless geographic database.*

1 Introduction

The deficiencies of using map sheets in Geographic Information Systems are well-known and have been described by several authors [5, 10]. The obvious answer to these deficiencies is a *seamless* or sheetless database. A seamless database is made possible in an interactive environment by using some form of multi-dimensional indexing, e.g. the R-tree [15] or the KD2B-tree [35]. It turns out that the *integrated* storage of multi-scale (*scaleless*) data in a spatial indexing structure forms the bottleneck in the design of a seamless, scaleless database [14]. A first approach might be to define a discrete number of levels of detail and store them separately each with its own spatial indexing structure. Though fast enough for interactive applications, this solution is not particularly elegant. It introduces redundancy because some

*A part of this work was done while the author was at the Department of Computer Science, University of Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands.

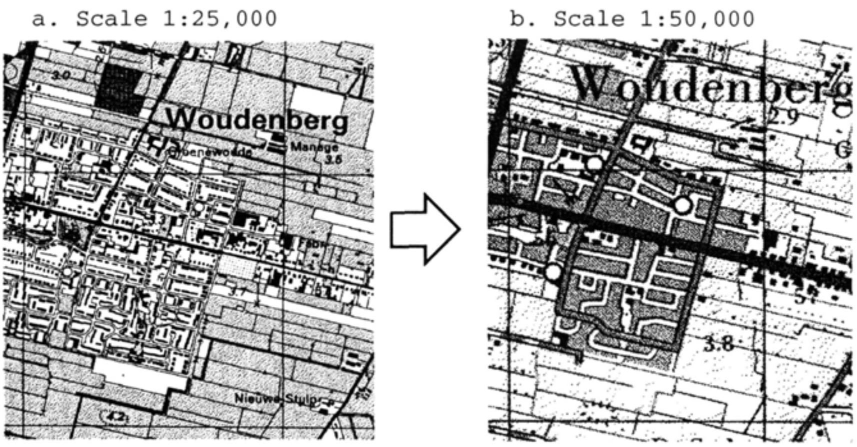


Figure 1: The Map Generalization Process

objects have to be stored at several levels. Apart from the increased memory usage, another drawback is that the data must be kept explicitly consistent. If an object is edited at one level, its "counter part" at the other levels must be updated as well. In order to avoid these problems we should try to design a storage structure that offers both spatial capabilities and multiple detail levels in an integrated manner: a *reactive data structure*. Two spatial data structures, that provide some limited facilities for multiple detail levels, are known: the Field-tree [9, 11] and the reactive BSP-tree [33, 34]. However, these are not fully dynamic.

First, we will discuss some of the fundamental problems associated with detail levels in a multi-scale database. The concept of multiple detail levels can not be defined as sharply as that of spatial searching. It is related to one of the main topics in cartographic research: *map generalization*; that is, to derive small scale maps (large regions) from large scale maps (small regions). Figure 1 illustrates the generalization process by showing the same part of a 1:25,000 map and of an enlarged 1:50,000 map. A number of generalization techniques for geographic entities have been developed and described in the literature [26, 30, 31]:

- simplification (e.g. line generalization);
- combination (aggregate geometrically or thematically);
- symbolization (e.g. from polygon to polyline or point);
- selection (eliminate, delete);
- exaggeration (enlarge); and
- displacement (move).

Unlike spatial searching, which is a pure geometric/topologic problem, map generalization is application dependent. The generalization techniques are categorized into two groups [23, 26]: geometric and conceptual generalization. In *geometric generalization* the basic graphic representation type remains the same, but is, for example, enlarged. This is not the case in *conceptual generalization* in which the

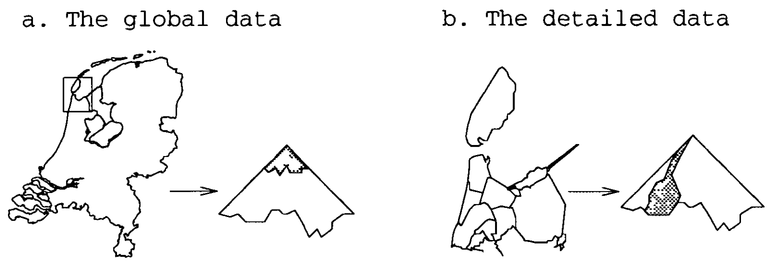


Figure 2: The Place of Global and Detailed Data

representation changes, e.g. change a river from a polygon into a polyline type of representation.

Generalization is a complex process of which some parts, e.g. line generalization [21, 22], are well suited to be performed by a computer and others are more difficult. Nickerson [25] shows that very good results can be achieved with a rule based expert system for generalization of maps that consist of linear features. Shea and McMaster [31] give guidelines for when and how to generalize. Müller [24] also applies a rule based system for selection (or its counterpart: elimination) of geographic entities. Brassel and Weibel [4] present a framework for automated map generalization. Mark [20], Müller [23], and Richardson [29] all state that the nature of the phenomenon must be taken into account during the generalization in addition to the more traditional guidelines, such as: the graphic representation (e.g. number of points used to draw a line) and the map density. This means that it is possible that a different generalization technique is required for a line representing a road than for a line representing a river. It is important to note that the spatial data structure with detail levels, presented in this paper, is only used to store the results of the generalization process.

The *guideline* that important objects must be stored in the higher levels of the tree, is the starting point for the design of the Reactive-tree. This *guideline* was derived during the development of the reactive BSP-tree [33, 34] and is illustrated in Figure 2: the global data are stored in the top levels of the tree (gray area in Figure 2a) and the detailed data of the selected region are stored in the lower levels of the tree (Figure 2b) in nodes which are "quite close" to each other. The Reactive-tree is an index structure, which supports geometric searching at different levels of importance. The properties of the Reactive-tree are described in Section 2, together with a straightforward Search algorithm. Insert and Delete algorithms are given in the subsequent section. Support for the generalization technique *simplification* is provided by representing polygonal or polyline objects by a Binary Line Generalization-tree, see Section 4. Support for the generalization techniques *aggregation* and *symbolization* is discussed in Section 5. In Section 6 the Alternative Reactive-tree is presented, not based on the *guideline* stated above. This paper is concluded with an evaluation of the presented structures.

2 The Properties of the Reactive-tree

In the following subsection, it is argued that *importance values* associated with objects, are required. The two subsequent subsections give an introduction to the Reactive-tree and a formal description of its properties, respectively. The last subsection describes a geometric Search algorithm, which takes the required importance level into account.

2.1 Importance Values

Generalization is, stated simply, the process of creating small scale (coarse) maps out of detailed large scale maps. One aspect of this process is the removal of unimportant and often, but not necessarily, small objects. This can be repeated a number of times, each time resulting in a smaller scale map with fewer objects in a fixed region. Each object is assigned a logical *importance value*, a natural number, in agreement with the smallest scale on which it is still present. Less important objects get low values, more important objects get high values. The use of importance values for the selection of objects was first published by Frank [8].

Which objects are important is depends on the application. In many applications a natural hierarchy is already present. In the case of, for example, a road map these are: highways, major four-lane roads, two-lane roads, undivided roads, and dirt roads. Another example can be found in WDB II [12] where lakes, rivers, and canals are classified into several groups of importance. Typically, the number of levels is between five and ten, depending on the size and type of the geographic data set. In a reasonable distribution the number of objects having a certain importance is one or two orders of magnitude larger than the number of objects at the next higher importance level; a so called, *hierarchical distribution.*

2.2 Introduction to the Reactive-tree

Several existing geometric data structures are suited to be adapted for the inclusion of objects with different importance values, for example the R-tree [15], the Sphere-tree, and the dynamic KD2B-tree [35]. In this paper, the Reactive-tree is based on the R-tree, because the R-tree is the best known structure. However, if orientation insensitivity is important, then one of the other structures mentioned must be used. The Reactive-tree is a multi-way tree in which, normally, each node contains a number of entries. There are two types of entries: *object-entries* and *tree-entries.* The internal nodes may contain both, in contrast to the R-tree. The leaf nodes of the Reactive-tree contain only object-entries. An object-entry has the form

(MBR, imp-value, object-id)

where *MBR* is the minimal bounding rectangle, *imp-value* is a natural number that indicates the importance, and *object-id* contains a reference to the object. A tree-entry has the form

(MBR, imp-value, child-pointer)

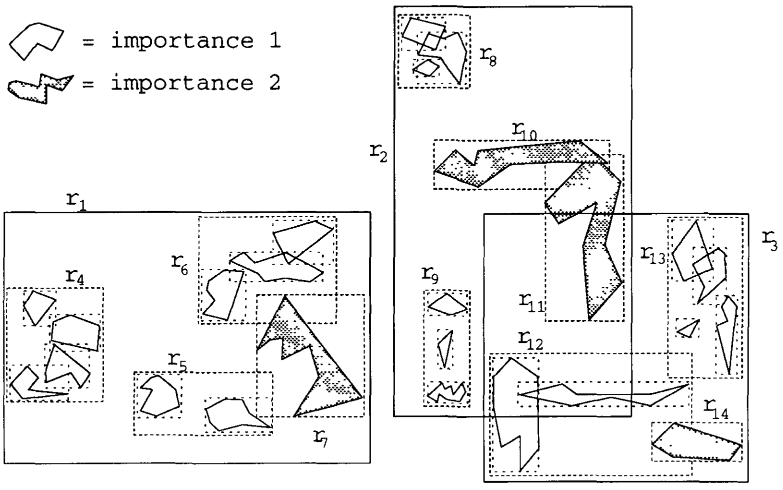


Figure 3: The Scene and the Rectangles of the Reactive-tree

where *child-pointer* contains a reference to a subtree. In this case *MBR* is the minimal bounding rectangle of the whole subtree and *imp-value* is the importance of the child-node incremented by 1. The importance of a node is defined as the importance of its entries. Note that the size of a tree-entry is the same as that of an object-entry. When one bit in the *object-id/child-pointer* is used to discriminate between the two entry types, then there is no physical difference between them in the implementation. Each node of the Reactive-tree corresponds to one disk page. Just as in the R-tree, M indicates the maximum number of entries that will fit in one node, and $m \leq \lceil M/2 \rceil$ is the minimum number of entries. Assume that the page size is 1024, then M is 48 in a realistic implementation.

2.3 Defining Properties

In this subsection the defining properties of the Reactive-tree are presented. The fact that the empty tree satisfies these properties and that the Insert and Delete algorithms given in Section 3 do not destroy them, guarantees that a Reactive-tree always exists. The Reactive-tree satisfies the following properties:

1. For each object-entry *(MBR, imp-value, object-id)*, *MBR* is the smallest axes-parallel rectangle that geometrically contains the represented object of importance *imp-value*.
2. For each tree-entry *(MBR, imp-value, child-pointer)*, *MBR* is the smallest axes-parallel rectangle that geometrically contains all rectangles in the child node and *imp-value* is the importance of the child-node incremented by 1.
3. All the entries contained in nodes on the same level are of equal importance, and more important entries are stored at higher levels.
4. Every node contains between m and M object-entries and/or tree-entries, unless it has no brothers (a *pseudo-root*).

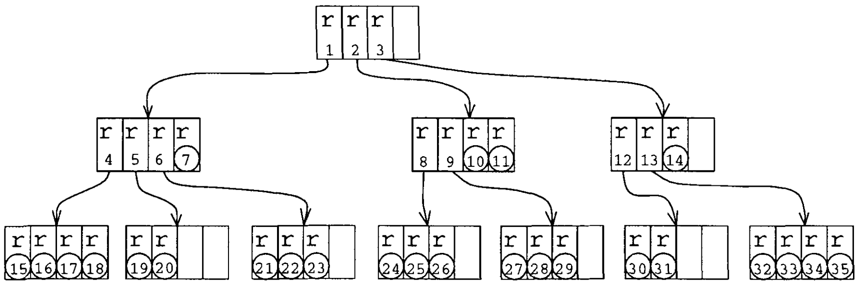


Figure 4: The Reactive-tree

5. The root contains at least 2 entries unless it is a leaf.

It is not difficult to see that the least important object-entries of the whole data set are always contained in leaf nodes on the same level. In contrast to the R-tree, leaf nodes may also occur at higher levels, due to the more complicated balancing criteria which are required by the multiple importance levels; see properties 3, 4, and 5. Further, these properties imply that in an internal node containing both object-entries and tree-entries, the importance of the tree-entries is the same as the importance of the object-entries. Figure 3 shows a scene with objects of two importance levels: objects of importance 1 are drawn in white and the objects of importance 2 are drawn in grey. This figure also shows the corresponding rectangles as used in the Reactive-tree. The object-entries in the Reactive-tree are marked with a circle in Figure 4. The importance of the root node is 3, and the importance of the leaf nodes is 1.

2.4 Geometric Searching with Detail Levels

The further one zooms in, the more tree levels must be addressed. Roughly stated, during map generation based on a selection from the Reactive-tree, one should try to choose the required importance value such that a constant number of objects will be selected. This means that if the required region is large only the more important objects should be selected and if the required region is small, then the less important objects must be selected also. The recursive *Search* algorithm to report all object-entries that have at least importance *imp* and whose *MBRs* overlap search region S, is invoked with the root of the Reactive-tree as current node:

1. If the importance of the current node N is less than *imp*, then there are no qualifying records in this node or in one of its subtrees.
2. If the importance of the current node N is greater or equal to *imp*, then report all object-entries in this node that overlap S.
3. If the importance of the current node N is greater than *imp*, then also invoke the Search algorithm for the subtrees that correspond to tree-entries that overlap S.

3 Insert and Delete Entry Algorithms

The Search algorithm is the easy part of the implementation of the Reactive-tree. The hard part is presented by Insert and Delete algorithms that do not destroy the properties of the Reactive-tree. In the implementation presented here, there is exactly one level in the Reactive-tree for each importance value, in the range from *min_imp* to *max_imp*, where *min_imp* and *max_imp* correspond to the least and to the most important object, respectively. If necessary, there may be one or more tree levels on top of this, which correspond to importance levels *max_imp* + 1 and higher. Then the top level nodes contain tree-entries only. Assume that *tree_imp* $\geq$ *max_imp* is the importance of the root of the Reactive-tree, then the height of the tree is *tree_imp* +1 - *min_imp*. The values of *min_imp* and *tree_imp* are stored in global variables. In the algorithms described below, the trivial aspects of maintaining the proper values of these variables are often ignored. Because of the direct relationship between the importance and the level of a node in the Reactive-tree of this implementation, the *imp_value* may be omitted in both the object-entry and the tree-entry.

3.1 Insert Entry

The Insert algorithm described below does not deal with the special cases: empty tree and the insertion of an entry with importance greater than *tree_imp*. Solutions for both are easy to implement and set the global variable *tree_imp* to the proper value. The *Insert* algorithm to insert a new entry E of importance *E_imp* in the Reactive-tree:

1. Descend the tree to find the node, that will be called N, by recursively choosing the best tree-entry until a node of importance *E_imp* or a leaf is reached. The best tree-entry is defined as the entry that requires the smallest enlargement of its MBR to cover E. While moving down the tree, adjust the MBRs of the chosen tree-entries on the path from the root to node N.
2. In the special case that node N is a leaf and the importance *N_imp* is greater than *E_imp*, a linear path (with length *N_imp* $-$ *E_imp*) of nodes is created from node N to the new entry. Each node in this path contains only one entry. This is allowed, because these are all pseudo-roots.
3. Insert the (path to) new entry E in node N. If overflow occurs split the node into nodes N and N' and update the parent. In case the parent overflows as well, propagate the node-split upward.
4. If the node-split propagation causes the root to split, increment *tree_imp* by 1 and create a new root whose children are the two resulting nodes.

The node splitting in step 3 is analogous to the node splitting in the R-tree. A disadvantage of the Reactive-tree is the possible occurrence of pseudo-roots. These may cause excessive memory usage in case of a "weird" distribution of the number of objects per importance level; e.g. there are more important objects than unimportant objects.

3.2 Delete Entry

An existing object is deleted by the *Delete* algorithm:

1. Find the node N containing the object-entry, using its MBR.
2. Remove the object-entry from node N. If underflow occurs, then the entries of the under-full node have to be saved in a temporary structure and the node N is removed. In case the parent also becomes under-full, repeat this process. It is possible that the node-underflow continues until the root is reached and in that case *tree_imp* is decremented.
3. Adjust the MBRs of all tree-entries on the path from the removed object-entry back to the root.
4. If underflow has occurred, re-insert all saved entries on the proper level in the Reactive-tree by using the Insert algorithm.

There are three types of underflow in the Reactive-tree: the root contains 1 tree-entry only, a pseudo-root contains 0 entries, or one of the other nodes contains $m-1$ entries. The temporary structure may contain object-entries and tree-entries of different importance levels.

4 The Binary Line Generalization-tree

Selection, as supported by the Reactive-tree, can assure that only global and important polylines (or polygons) are selected out of a large-scale geographic data set, when a small-scale map (large regions) has to be displayed. However, without specific measures, these polylines are drawn with too much detail, because all points that define the polyline are used. This detail will be lost on this small-scale due to the limited resolution of the display. Also the drawing will take an unnecessary long period of time. It is better to use fewer points. This can be achieved by the *k-th point algorithm*, which only uses every k-th point of the original polyline for drawing. The first and the last points of a polyline are always used. This is to ensure that the polylines remain connected to each other in the nodes of a topologic data structure [3, 27]. This algorithm can be performed "on the fly" because it is very simple. The k can be adjusted to suit the specified scale. However, this method has some disadvantages:

- The shape of the polyline is not optimally represented. Some of the line characteristics may be lost if the original polylines contain very sharp bends or long straight line segments.
- If two neighboring administrative units are filled, for example, in case of a choropleth, and the k-th point algorithm is applied on the contour, then these polygons may not fit. The contour contains the re-numbered points of several polylines.

Therefore, a better line generalization algorithm has to be used, for instance the *Douglas-Peucker algorithm* [6]. Duda and Hart [7] describe an algorithm similar to the Douglas-Peucker algorithm and call it the "iterative end-point fit" method. Both references date back to 1973. A slightly earlier publication is given by Ramer [28] in 1972. These types of algorithms are time consuming, so it is wise to compute

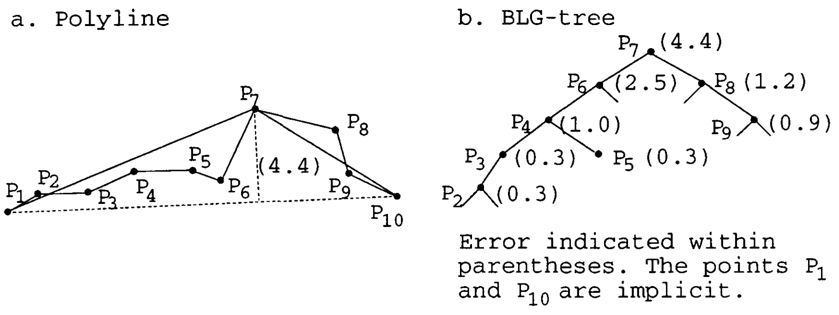


Figure 5: A Polyline and its BLG-tree

the generalization information for each polyline in a pre-processing step. The result is stored in, for instance, a Multi-scale Line Tree [17, 18]. The disadvantages of the Multi-scale Line Tree have already been discussed in [37]: it introduces a discrete number of detail levels and the number of children per node is not fixed. Strip trees [1] and Arc trees [13] are binary trees that represent curves (in a 2D-plane) in a hierarchical manner with increasing accuracy in the lower levels of the tree. These data structures are designed for arbitrary curves and not for simple polylines. Therefore, we introduce a new data structure that combines the good properties of the structures mentioned. We call this the *Binary Line Generalization-tree* (BLG-tree).

The BLG-tree stores the result of the Douglas-Peucker algorithm in a binary tree. The original polyline consists of the points p_1 through p_n The most coarse approximation of this polyline is the line segment $[p_1, p_n]$. The point of the original polyline, that has the largest distance to this line segment, determines the error for this approximation. Assume that this is point p_k with distance d, see Figure 5a. p_k and d are stored in the root of the BLG-tree, which represents the line segment $[p_1, p_n]$. The next approximation is formed by the two line segments $[p_1, p_k]$ and $[p_k, p_n]$. The root of the BLG-tree contains two pointers to the nodes that correspond with these line segments. In the "normal" situation this is a more accurate representation.

The line segments $[p_1, p_k]$ and $[p_k, p_n]$ can be treated in the same manner with respect to their part of the original polyline as the line segment $[p_1, p_n]$ to the whole polyline. Again, the error of the approximation by a line segment can be determined by the point with the largest distance. And again, this point and distance are stored in a node of the tree which represents a line segment. This process is repeated until the error (distance) is 0. If the original polyline does not contain three or more collinear points, the BLG-tree will contain all points of that polyline. It incorporates an exact representation of the original polyline. The BLG-tree is a static structure with respect to inserting, deleting and changing points that define the original polyline. The BLG-tree of the polyline of Figure 5a is shown in Figure 5b. In most cases, the distance values stored in the nodes will become smaller when descending the tree. Unfortunately, this is not always the case, as shown in Figure 6. It is not a monotonically decreasing series of values.

The BLG-tree is used during the display of a polyline or polygon at a certain scale. One can determine the maximum error that is allowed at this scale and the primitive

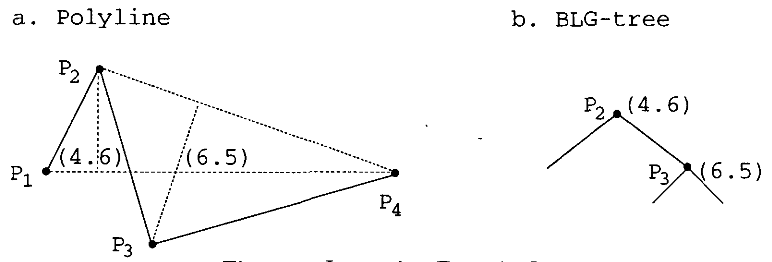


Figure 6: Increasing Error in BLG-tree

is *simplified* and a good graphic representation is obtained. During traversal of the tree, one does not have to go any deeper in the tree once the required accuracy is met. The BLG-tree can also be used for other purposes, for example (further details can be found in [37]):

- Estimating the area of a region enclosed by a number of polylines.
- Estimating the intersection(s) of two polylines. This is a useful operation during the calculation of a map overlay (polygon overlay).

Note that the BLG-tree is most useful for polylines and polygons defined by a large number of points. For a small number of points, "on the fly" execution of the Douglas-Peucker [6] algorithm may be more efficient. For polylines that are somewhere in between, another alternative might be interesting. Assign a value to each point to decide whether the point is used when displaying the polyline at a certain scale. This simple linear structure is probably fast enough for the medium sized polyline.

5 Support for Other Generalization Techniques

The Reactive-tree and the BLG-tree reflect only a part of the map generalization process: selection and simplification. A truly reactive data structure also deals with other aspects of the generalization process. In this section two more aspects are discussed: symbolization, and aggregation. These terms may be confusing in the context of the Reactive-tree, because the tree is usually described "top-down" (starting with the most important objects) and map generalization is usually described "bottom-up" (starting at the most detailed level). The two generalization techniques are incorporated in the reactive data structure by considering objects not as a simple list of coordinates, but as more complex structures. In practice, this can be implemented very well by using an object-oriented programming language, such as Procol [19, 32, 36, 37].

Symbolization changes the basic representation of a geographic entity, for example, a polygon is replaced by a polyline or point on a smaller scale map. Besides the coordinates of the polygon, the object structure contains a second representation in the form of a polyline or point. Associated with each representation is a resolution range which indicates where it is valid. An example of the application of the symbolization technique is a city which is depicted on a small scale map as a dot and

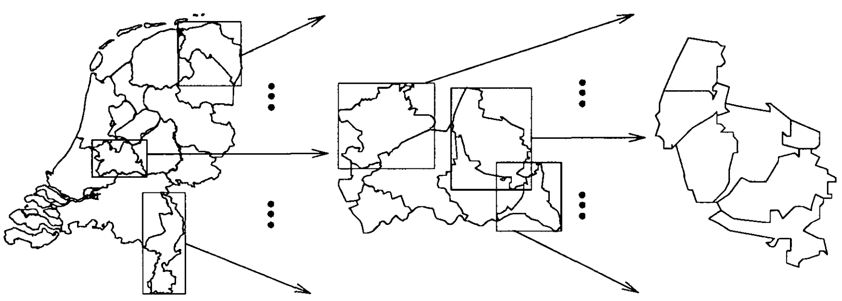


Figure 7: A Large Object is Composed of Several Small Objects

on a large scale map as a polygon.

The last generalization technique included in the reactive data structure is *aggregation*, that is the combination of several small objects into one large object. From the "top-down hierarchical tree" point of view, a large object is composed of several small objects; see Figure 7. The geometric description of the large object and the geometric descriptions of the small objects are all stored, because there is no simple relationship between them. The large object is some kind of "hull" around the small objects, see Figure 7. Usually, a bounding box around the small objects is a sufficient "geometric search structure", because the number of small objects is limited. However, if the number of small objects combined in one large object is quite large, then a R-subtree may be used.

Aggregation is used, for example, in the map of administrative units in The Netherlands [37]. Several municipalities are grouped into one larger economic geographic region (EGR), EGRs are grouped into a nodal region, nodal regions are grouped in a province, and so on. Another approach to this case is to consider the boundaries as starting point of the design, instead of the regions. In that case selection is the appropriate generalization technique and the Reactive-tree can be used without additional structures.

6 An Alternative Reactive-tree

In this section a reactive data structure is presented, which is not based on the guideline that important objects must be stored in the higher levels of the tree. The advantage of the *Alternative Reactive-tree* over the Reactive-tree is that it does not assume a hierarchical distribution of the number of objects over the importance levels.

The 2D Alternative Reactive-tree is based on a 3D R-tree. The 3D MBR of a 2D object with importance *imp* is defined by its 2D MBR and its extents in the third dimension are from *imp* and to *imp*+δ, where δ is a positive real number, so an object corresponds to a block with non-zero contents (except for point objects). Figure 8 depicts the 3D MBRs of a number of 2D objects at two different importance levels. When the parameter δ is chosen very small, e.g. 0.01, the Alternative Reactive-tree tries to group the objects that belong to the same importance level. This can be explained by the fact that there is a heavy penalty on the inclusion of an object with

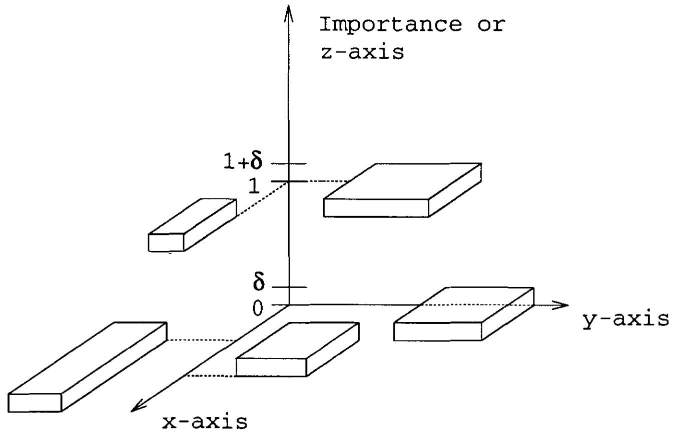


Figure 8: The 3D MBRs of the Alternative Reactive-tree

another importance value, as the volume of the 3D MBR will increase by at least a factor $(1 + \delta)/\delta$. The larger δ becomes, the less the penalty, and the more likely it is that objects of different importance are grouped, and the Alternative Reactive-tree behaves more like a normal 2D R-tree. In any case, all objects, important and unimportant, are stored in leaf nodes on the same level.

The Alternative Reactive-tree can be generalized to support objects with general *labels* instead of the hierarchical importance values. This enables queries such as "Select all *capital* cities in region *R*." The label *capital* is associated with some of the geographic objects, by inserting these entries into the tree. A Geographic object may be associated with more labels by inserting more entries for the same object. In the implementation, label corresponds to a numeric value. By choosing certain values for these labels and for δ, possible coherence between labels may be exploited. This is what is actually done in the 2D Alternative Reactive-tree for hierarchically distributed data.

7 Discussion

This paper described the first fully dynamic and reactive data structure. It was presented as a 2D structure, but 3D and higher dimensional variants are possible. Note that this has nothing to do with the use of a 3D R-tree for the 2D Alternative Reactive-tree. The Reactive-tree and the Alternative Reactive-tree have been implemented in C++ on a Sun 3/60. Two large data sets have been used to test the reactive structures: WDB II [12] and the map of administrative units in The Netherlands. Both performance tests showed the advantage of the *selection* based on importance level and geometric position. Displaying the whole map area at interactive speed was possible, in contrast to the situation where the normal R-tree was used, which also showed a lot of annoying details. The additional structures for the support of simplification, symbolization, and aggregation are currently being implemented. Future performance tests depend on the availability of digital maps with generalization information.

Two other generalization techniques where not discussed: exaggeration and displacement. *Exaggeration* seems easy to include, because it is a simple enlargement of an aspect of the graphic representation of one object, e.g. the line width. However, the enlargement of linear features may cause other features to be covered and they must therefore be *displaced.* Exaggeration and displacement are difficult to handle, because multiple objects have to be considered. An ad hoc solution is to associate an explicit set of tuples *(displacement, map-scale-range)* with each object that has to be displaced and a set of tuples *(enlargement, map-scale-range)* with each object that has to be enlarged. Further research is required in order to develop more elegant solutions.

Very recently, another reactive data structure has been proposed by Becker and Widmayer [2]. The *Priority Rectangle File* (PR-file, based on the R-file [16]) forms the backbone of their structure. A significant common characteristic of the PR-file and the Reactive-tree is that, in general, both store more important objects in higher levels. A few differences of the PR-file, compared to the Reactive-tree, are: objects of equal importance (priority) are not necessarily on the same level, and object-entries and tree-entries can not be stored in the same node.

Finally, other Reactive-trees should be considered which are able to deal efficiently with a non-hierarchical distribution of the number of objects over the importance levels, whilst sticking to the guideline that important objects are to be stored in the higher levels of the tree. This might be realized by changing the properties in such a manner that one tree level is allowed to contain multiple importance levels, but it is not (yet) clear how the Insert and Delete algorithms should be modified. This is subject to further research.

References

[1] Dana H. Ballard. Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5):310–321, May 1981.

[2] Bruno Becker and Peter Widmayer. Spatial priority search: An access technique for scaleless maps. Technical report, Institut für Informatik, Universität Freiburg, June 1990.

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

[4] Kurt E. Brassel and Robert Weibel. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*, 2(3):229–244, 1988.

[5] Nicholas R. Chrisman. Deficiencies of sheets and tiles: Building sheetless databases. *International Journal of Geographical Information Systems*, 4(2):157–167, 1990.

[6] 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.

[7] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[8] A. Frank. Application of DBMS to land information systems. In *Proceedings of the Seventh International Conference on Very Large Data Bases*, pages 448–453, 1981.

[9] André Frank. Storage methods for space related data: The Field-tree. Technical Report Bericht Nr. 71, Eidgenössische Technische Hochschule Zürich, June 1983.

[10] Andrew U. Frank. Requirements for a database management system for a GIS. *Photogrammetric Engineering and Remote Sensing*, 54(11):1557–1564, November 1988.

[11] Andrew U. Frank and Renato Barrera. The Field-tree: A data structure for Geographic Information System. In *Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, California*, pages 29–44. Lecture Notes in Computer Science 409, Springer Verlag, July 1989.

[12] Alexander J. Gorny and Russ Carter. World Data Bank II, General users guide. Technical report, U.S. Central Intelligence Agency, January 1987.

[13] Oliver Günther. *Efficient Structures for Geometric Data Management.* Number 337 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1988.

[14] Stephen C. Guptill. Speculations on seamless, scaleless cartographic data bases. In *Auto-Carto 9, Baltimore*, pages 436–443, April 1989.

[15] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD*, 13:47–57, 1984.

[16] Andreas Hutflesz, Hans-Werner Six, and Peter Widmayer. The R-file: An efficient access structure for proximity queries. In *Proceedings IEEE Sixth International Conference on Data Engineering, Los Angeles, California*, pages 372–379, February 1990.

[17] Christopher B. Jones and Ian M. Abraham. Design considerations for a scale-independent cartographic database. In *Proceedings 2nd International Symposium on Spatial Data Handling, Seattle*, pages 348–398, 1986.

[18] Christopher B. Jones and Ian M. Abraham. Line generalization in a global cartographic database. *Cartographica*, 24(3):32–45, 1987.

[19] Chris Laffra and Peter van Oosterom. Persistent graphical objects. In *Eurographics Workshop on Object Oriented Graphics*, June 1990.

[20] David M. Mark. Conceptual basis for geographic line generalization. In *Auto-Carto 9, Baltimore*, pages 68–77, April 1989.

[21] Robert B. McMaster. Automated line generalization. *Cartographica*, 24(2):74–111, 1987.

[22] J.-C. Müller. Optimum point density and compaction rates for the representation of geographic lines. In *Auto-Carto 8*, pages 221–230, 1987.

[23] Jean-Claude Müller. Rule based generalization: Potentials and impediments. In *4th International Symposium on Spatial Data Handling, Zürich, Switzerland*, pages 317–334, July 1990.

[24] Jean-Claude Müller. Rule based selection for small scale map generalization. Technical report, ITC Enschede, The Netherlands, 1990.

[25] Bradford G. Nickerson. *Automatic Cartographic Generalization For Linear Features.* PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, April 1987.

[26] F.J. Ormeling and M.J. Kraak. *Kartografie: Ontwerp, productie en gebruik van kaarten* (in Dutch). Delftse Universitaire Pers, 1987.

[27] Thomas K. Peucker and Nicholas Chrisman. Cartographic data structures. *The American Cartographer*, 2(1):55–69, 1975.

[28] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[29] Diane E. Richardson. Database design considerations for rule-based map feature selection. *ITC Journal*, 2:165–171, 1988.

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

[31] K. Stuart Shea and Robert B. McMaster. Cartographic generalization in a digital environment: When and how to generalize. In *Auto-Carto 9, Baltimore*, pages 56–67, April 1989.

[32] Jan van den Bos and Chris Laffra. Procol - A parallel object language with protocols. In *OOPSLA '89, New Orleans*, pages 95–102, October 1989.

[33] Peter van Oosterom. A reactive data structure for Geographic Information Systems. In *Auto-Carto 9, Baltimore*, pages 665–674, April 1989.

[34] Peter van Oosterom. A modified binary space partitioning tree for Geographic Information Systems. *International Journal of Geographical Information Systems*, 4(2):133–146, 1990.

[35] Peter van Oosterom and Eric Claassen. Orientation insensitive indexing methods for geometric objects. In *4th International Symposium on Spatial Data Handling, Zürich, Switzerland*, pages 1016–1029, July 1990.

[36] Peter van Oosterom and Chris Laffra. Persistent graphical objects in Procol. In *TOOLS '90, Paris*, pages 271–283, June 1990.

[37] Peter van Oosterom and Jan van den Bos. An object-oriented approach to the design of Geographic Information Systems. *Computers & Graphics*, 13(4):409–418, 1989.

INTEGRATION OF SPATIAL OBJECTS IN A GIS

Richard G. Newell, Mark Easterfield & David G. Theriault
Smallworld Systems Ltd,
8-9 Bridge Street,
Cambridge, England
CB2 1UA

ABSTRACT

A GIS is distinguished from other database systems in a number of respects, particularly in the requirement to handle spatial objects with extent. Whereas a common approach is to treat "geometry" and "attributes" separately, a more integrated approach is to treat the spatial aspect as but one part of an integrated data model which accommodates all objects and their attributes in a seamless manner. Spatial objects differ from other object attributes in that they usually have extent and therefore efficient retrievals cannot be achieved by the mechanisms implemented within database systems on their own. This paper addresses the problem of implementing efficient spatial retrieval methods within an integrated object data model. An improved quadtree mechanism for clustering objects on disk is also described.

INTRODUCTION

In recent years, the problem of organising large numbers of spatial objects in a database has become much better understood. It is now up to the system implementors to apply the known methods in real systems. However, although there are many algorithms described for indexing and storing spatial objects, there is little published information on how to apply the algorithms within the context of a complete integrated database system which would support a GIS. In particular, there is an almost total lack of clear descriptions of implementations within the realm of the relational model, although papers have been published which seem to get good results using this approach (Abel 1983 and 1984, Bundock 1987).

Older systems use a proprietary structure of sheets or tiles to organise their spatial data, but this leads to serious problems in large systems. A modern approach is to implement a spatial database which is logically seamless. Some systems separate out the geometric objects into a separate proprietary database which is specifically built for fast spatial retrievals, and then this is linked somehow to a database of attributes. It is our contention that this does not meet any criteria of integration. This does of course beg the question of what to do about the integration of spatially located data that is already committed to institutional databases and which needs to be accommodated within a GIS. We have no elegant answer to this problem.

Currently available relational database systems come under a lot of criticism for use with spatial data, on account of their apparent poor performance. However, in our view, the jury is still out on this issue, as, if appropriate data models are used, then acceptable spatial performance seems to be achievable (Abel 1983 and 1984), but this depends on the use to be made of the GIS. For relational databases, there are more serious issues than this to overcome, especially the management of long transactions and versioning (Easterfield et al 1990).

It is significant however, that most commercial systems that use a commercial relational database system for holding spatial data employ a system of check out into a single user proprietary database before work starts, and they may even go one step further to employ a display file to gain adequate graphics performance.

We have been researching the implementation of fast spatial retrieval algorithms within the context of a version managed tabular database, which uses an interactive object oriented language for its development and customisation environment (Chance et al 1990, Easterfield et al 1990). Our approach has been to implement spatial retrieval methods in the system by using the normal indexing methods of the tabular database, without any ad hoc structures showing through to the system customiser and developer. We neither employ check out for reasons of handling multiple users nor do we need to employ a display file to achieve good performance.

HOW TO ACHIEVE FAST SPATIAL ACCESS

The nub of all spatial access algorithms (e.g. range trees, quadtrees, field trees, k-d trees etc) seems to be the same, in that if one can organise one's data somehow in the form of a tree structure, where each node of the tree corresponds to a subset of the area covered by its parent node, then candidate objects inside an area or surrounding a point can be found quickly. Such algorithms can retrieve an object in a time proportional to the logarithm of the number of objects in the database.

One approach is to provide an external spatial index into a database of objects which is not spatially organised. If one is retrieving many objects within say a rectangle, then the candidate objects can be identified very quickly, but retrieving the objects themselves is like using a pair of tweezers to extract each one from disk. The logarithmic behaviour still applies, but the constant term is very large because of disk seek time.

Thus, to gain the full benefit, the actual object data itself needs to be organised spatially on disk, by clustering the data, so that one can at least use a "shovel" to retrieve objects (bulk retrieval), instead of a pair of "tweezers" (random retrievals).

Certain methods of spatial indexing are structured so that each object is contained once only in the index. Other approaches duplicate an object's entry in the index, based on sub-ranges of the total object. This has the advantage that candidate objects are more likely to be relevant, but the disadvantage that duplicates must be eliminated. (Abel 1983)

There does not seem to be a great performance difference between the many methods of indexing and clustering that have been described in the literature (Smith and Gao 1990, Kriegel et al 1990). The message is, just do anything in a tree structure and you will get most of the benefit, the rest is just tuning.

However, we are rather concerned with implementing such mechanisms within an overall integrated data model, where the peculiarities of particular methods are hidden, because if they are not, the complexity and cost of development is increased. In addition, as new spatial indexing mechanisms are discovered, these should be implementable independently of the overall data model.

SPATIAL KEYS IN TABULAR DATABASES

It is well known that it is possible to encode the size and position of an object in a unique spatial key, so that objects which are close to each other in space generate similar keys. Further, if objects with similar keys are stored at similar locations on disk, then the number of disk accesses required to retrieve objects can be greatly minimised.

Some methods lend themselves easily to the generation of a spatial key, such as a quadtree index and its many close variations (Samet 1989). However, mechanisms such as range trees preclude this approach, indeed the actual structure produced depends on the order in which the database is created.

We have not investigated methods such as range trees for clustering objects, because they are ad hoc and it seems to us difficult to hide the storage mechanisms behind an acceptable interface for system developers.

It is common in a tabular database that records with similar primary keys are close to each other on disk, especially if the fundamental storage mechanism is something like a B*tree. Thus, if the most significant part of the primary key of all spatial objects is a spatial key, then the desired effects of spatial ordering on disk can be achieved (Abel 1983 and 1984, Libera & Gosen 1986). Further, topological relationships between objects (e.g. represented by an association table) can also be arranged to have the same spatial keys as parts of their primary keys so these will become spatially clustered as well.

However, the approach has one potential drawback. Consider the problem of changing the geometry of such an object in a way that its spatial key changes. Thus, making some, possibly minor, geometric edit could result in a change to the primary key, i.e. identity, of the object, resulting in problems of maintaining the overall integrity of the database. Modification of a primary key effectively means deletion followed by re-insertion.

One might consider a storage mechanism where records are clustered according to a spatial key which is not a part of the primary key. While this would be satisfactory for extremely simple models (e.g. a single table of records with an auxiliary index for primary key retrievals) it would not cluster the records in the many association tables that exist in a richly connected data model.

A PRAGMATIC APPROACH

The idea of containing a spatial key within the identity of each object does not complicate other kinds of retrieval. As far as these procedures are concerned, a spatial key is no different from an ordinary key. Our pragmatic idea is that at the time an object is first created, we generate a spatial key as part of its unique identifier. The value of this identifier never changes from this point on, even if the location and geometric extent of the object is changed. The pragmatic part comes in that geometric edits are rare, and major changes in position or extent are even rarer. Thus the object rarely moves far in space, so why should it move far on disk? Thus an object's identity is a function of where it is born and we assume that it never moves far from its place of birth.

However, by doing this, spatial retrievals may become unreliable, because some objects which should be retrieved may be missed. Our solution to this is to have a single external spatial address table, with accurate spatial keys which are always maintained up-to-date, and it is this which is used to retrieve objects. The "sloppy" spatial key is no more than a clustering device, i.e. an accelerator to speed spatial search. In the worst case, if large parts of the geometry were modified significantly (e.g. following rectification), then the system may get slow, but it would still work correctly.

The method used to implement the index does not need to be the same as the method used to organise the clustering of the actual data so that, for example, a range tree index (Guttman 1984) could be used to index data which is clustered in a quadtree.

In our implementation, we use the same approach for both clustering the data on disk and for building the external index.

HOW TO GENERATE A SPATIAL KEY

As there seems to be general agreement that there is minimal difference in performance between the many tree-based approaches to spatial clustering, (Kriegel et al for example found differences of the order of 20% between the various methods that they investigated) then perhaps the next criterion could be to aim for simplicity. This therefore eliminates the range tree, because neither is it simple, nor is it easy to see how one generates a permanent, reproducible key from it. Smith and Gao found that methods based on B-trees were good on storage utilisation, insert speed and delete speed, but were inferior on search times. We suggest here a modification to the method of creating a key based on a linear quadtree which gives a worthwhile performance improvement without degrading the other performances, nor adding any undue complexity.

Now it is well known that point data can be incorporated in a Morton Sequence, which is directly equivalent to encoding the quadtree cell in which the point exists. A quadtree index is very good for encoding point data, because all points exist in the leaves of the tree (See figure 1) (see the MX quadtree in Samet 1989).

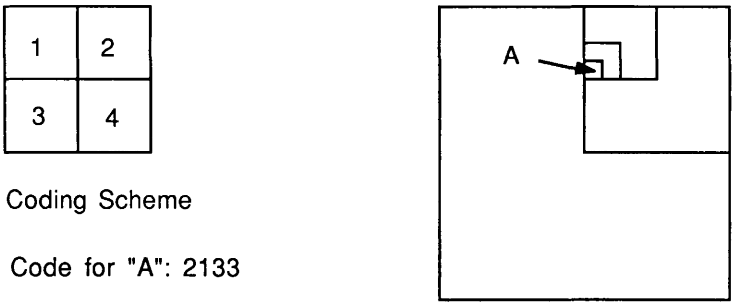
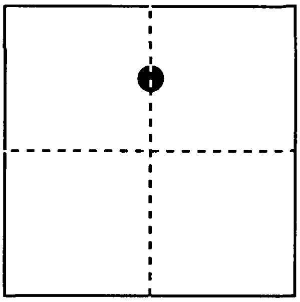


Figure 1: Quad-tree Encoding of Point Object

Encoding spatial objects with extent in a quadtree can also be done, but many objects will exist near the root (see MX-CIF Quadtree in Samet 1989 and Batty 1990), thereby leading to them being included in many retrievals when they are not in fact relevant. For large databases, this leads to a degradation in retrieval performance. Samet's book contains a number of schemes for getting around this problem by allowing an object to exist in more than one quadtree node. However, this is not suitable for clustering the object data itself.

In our earlier researches we had investigated solving this problem by using a key based on a nine-tree in which each square is divided into 9 equal sized overlapping squares each of which is a quarter of the size (half the dimension) of its parent. Although it had the desired effect of not populating the root, the tree is not well balanced and the retrieval strategy is more complex. This in fact was the basic approach behind Bundock's paper (Bundock 1987).

We include here a simpler solution to this problem because we have not seen it described elsewhere. The idea is based on the fact that most objects are very small compared to the world. So in order to avoid trapping objects near the root of the tree, the subdivision method is modified so that each quadtree cell is divided into 4 parts which overlap slightly, i.e each quadtree sub-square is slightly more than half the dimension of its parent square (See figure 2). The overlap never needs to be more than the size of the largest object in the database, and in practice can be less than this. The optimum overlap depends on some statistic of the object size, such as the mean object size times a factor.



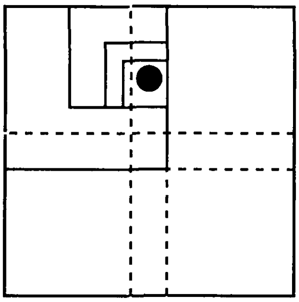


Figure 2: Quad-tree Encoding of Object with Extent

This slight modification to the simple quadtree key is no more complex to program, but does lead to worth while performance improvements for retrieving objects inside an area and in finding objects surrounding a point compared to the simple quadtree key mechanism.

A DATA MODEL FOR GIS

Figure 3 below illustrates graphically a simplified data model. It should be regarded as just a part of the complete model required for a GIS application. A large number of users' requirements for modelling their geometry and associated topology can be handled by a generic model of this form. Where users differ from one another is in the modelling of their own objects and interrelationships. The philosophy is not that geometric objects, such as polygons, have attributes, but that real world objects can be represented geometrically by such things as polygons. In this diagram, a line with an arrow signifies a one-to-many or many-to-one relationship and a line with an arrow at both ends signifies a many-to-many relationship. Of course, in a physical implementation, a many-to-many relationship is implemented by means of an intermediate table, which itself should also be spatially clustered.

The diagram should be read starting from the top. For example, a real world object, such as a wood is represented by an area, which may be made up of one or more polygons (these polygons may have resulted from intersections with other polygons in the same topological manifold). It is possible that each polygon may have one or more "islands" such as a lake (i.e in this case, a lake is an island). The lake area would of course share the same polygon as used by the wood area. Polygon boundaries are represented by a closed set of links, each one connecting exactly two nodes.

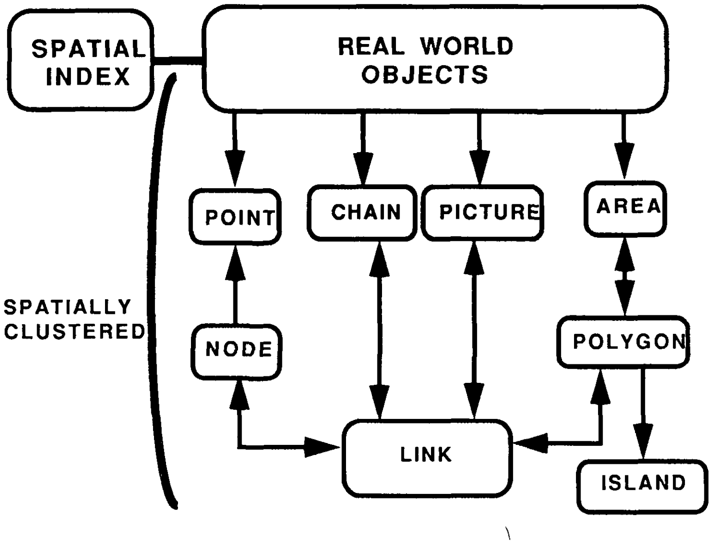


Figure 3: Spatially Indexed Topological Model

From the point of view of the system implementor, it is only the interrelationships of this model that he wishes to worry about, without himself being concerned with efficient spatial retrievals. However, if all entities in the model are identified by means of a key with spatial content, then the desired clustering of instances in each table will occur transparently. As it is common that objects which are topologically related are also spatially near to one another, disk accesses for topological queries should also be greatly reduced.

All that is needed in addition to this model is the external spatial index itself, which is merely a device for generating candidate keys for spatial retrieval. The point is that the spatial indexing method does not perturb the logical structure of the model.

In our implementation, the spatial index is a table, just like any other, which refers to real world objects, such as houses, lakes and utility pipe segments. A real world object may then have a number of different representations depending on such contexts as scale.

CONCLUSION

This paper has been concerned with the implementation of fast spatial indexing methods within the context of an integrated GIS data model. A design criterion has been to implement the spatial mechanisms without complicating other retrievals from the database. An approach is advocated based on a precise spatial index which can generate candidate keys within a tabular database whose primary keys contain a permanent, but "sloppy", spatial key. The external precise spatial index could be regarded as part of the data model, or could indeed be implemented as an entirely different mechanism. The method proposed for generating spatial keys is a minor improvement to the simple quadtree, which we have described here, because we have not seen it published elsewhere.

ACKNOWLEDGEMENTS

We are grateful to Mike Bundock for his helpful comments on an earlier draft of this paper.

REFERENCES

Abel, D.J. & Smith, J.L. (1983): A Data Structure and Query Algorithm Based on a Linear Key for a Rectangle Retrieval Problem, *Computer Vision, Graphics, and Image Processing 24,1, October 1983.*

Abel,D.J. & Smith, J.L. (1984): A Data Structure and Query Algorithm for a Database of Areal Entities, *The Australian Computer Journal, Vol 16, No 4.*

Batty, P. (1990): Exploiting Relational Database Technology in GIS, *Mapping Awareness magazine, Volume 4 No 6, July/August 1990.*

Bundock, M. (1987): An Integrated DBMS Approach to Geographical Information Systems, *Autocarto 8 Conference Proceedings, Baltimore, March/April 1987.*

Chance, A., Newell, R.G. & Theriault, D.G. (1990). An Object-Oriented GIS - Issues and Solutions, *EGIS '90 Conference Proceedings, Amsterdam, April 1990.*

Easterfield, M.E., Newell, R.G. & Theriault, D.G. (1990): Version Management in GIS - Applications and Techniques, *EGIS '90 Conference Proceedings, Amsterdam, April 1990.*

Guttman, A. (1984): R-trees: A Dynamic Index Structure for Spatial Searching, *Proceedings of ACM SIGMOD Conference on Management of Data, Boston, June 1984.*

Kriegel, H., Schiwietz, M., Schneider, R., Seeger, B. (1990): Performance Comparison of Point and Spatial Access Methods, *in Design and Implementation of Large Spatial Databases: Proceedings of the First Symposium SSD '89, Santa Barbara, July 1989.*

Libera, F.D. & Gosen, F. (1986): Using B-trees to Solve Geographic Range Queries, *The Computer Journal, Vol 29, No 2.*

Samet, H. (1989): The Design and Analysis of Spatial Data Structures, *Addison Wesley, 1990, ISBN 0-201-50255-0.*

Seeger, B. & Kriegel,H. (1988): Techniques for Design and Implementation of Efficient Spatial Access Methods, *Proceedings of the 14th VLDB Conference, Los Angeles, California, 1988.*

Smith, T. R. and Gao, P. (1990): Experimental Performance Evaluations on Spatial Access Methods *Proceedings of the 4th Spatial Data Handling Symposium, Zürich 1990, Vol 2, p991.*

ALTERNATE REPRESENTATIONS OF GEOGRAPHIC REALITY

Auto Carto 10
Panel Discussion

Organizers:
Stephen Guptill, U.S. Geological Survey
Scott Morehouse, E.S.R.I.

Participants:

Nicholas Chrisman, University of Washington
Michael Goodchild, University of California, Santa Barbara
Stephen Guptill, U.S. Geological Survey
Scott Morehouse, E.S.R.I.
François Salgé, IGN, France

THE DEBATE

Geographic information systems (GIS) are based on models of geographic reality. The functionality and utility of a GIS depends on a useful and correct data model. Data modeling involves the abstraction of reality as a number of objects or features, then defining these objects, their interrelationships, and behavior precisely.

The real world of geographical variation is infinitely complex and often uncertain, but must be represented digitally in a discrete, deterministic manner. Sometimes it is possible to define discrete features or objects in more or less rigorous fashion, but more often the digital representation is an abstraction of reality.

GIS databases present two very different views of reality. In one, geographical variation is represented by a set of layers, each of which records the pattern of one variable over the study area. If there are n layers, then n separate items of information are available for each and every point in the area. The variation in any one layer may be represented in numerous ways, including a raster of point samples, a set of nonoverlapping polygons, an irregular set of point samples, or a TIN.

In the second approach, we think of the world as a space populated by objects of various kinds - points, lines, and areas. Objects have attributes which serve to distinguish them from each other. Any point in the space may be empty, or occupied by one or more objects. GISs that take the layer view of the world often allow the user to populate a space with objects, but then insist that they be forced into the layer model.

However, much of the recent work on object-oriented design for geographic databases has emphasized the syntax of geographic features and deemphasized the need for defining useful spatial query, display, and analysis operators. It may be more important to design the data model around what geographic objects do, rather than what they are.

One contention is that the purpose of a GIS is not only to store and query a static schema of entities and relationships, but also to build new entities and relations dynamically. Why, therefore, define, capture, and store relationships which can be computed as needed? More importantly, why define and capture relationships between entities for which there is no clear functional requirement? Perhaps the goal of GIS data model design should be to develop the simplest model that works.

The layer/object debate is becoming as important to the current GIS industry as the earlier raster/vector debate, and carries a fundamental message. Whereas the raster/vector debate was over how to represent the contents of a map in a database, the layer/object debate is over how to represent and analyze the multivariate complexity of geographic reality.

The panelists will present a wide variety of views on the layer/object debate from the perspectives of academia, industry, and government. Each has been involved in the design, development, and use of geographic information systems and geographic databases. The commentary will help the audience to clarify their thoughts on this important topic.

WHAT'S THE BIG DEAL ABOUT GLOBAL HIERARCHICAL TESSELLATION?

Organizer: *Geoffrey Dutton*
150 Irving Street
Watertown MA 02172 USA
email: qtm@cup.portal.com

This panel will present basic information about hierarchical tessellations (HT's) as geographic data models, and provide specific details about a few prototype systems that are either hierarchical tessellations, global tessellations, or both. Participants will advocate, criticize and discuss these models, allowing members of the audience to compare, contrast and better understand the mechanics, rationales and significance of this emerging class of nonstandard spatial data models in the context of GIS. The panel has 7 members, most of whom are engaged in research in HT data modeling, with one or more drawn from the ranks of informed critics of such activity. The panelists are:

- Chairperson TBA
- Zi-Tan Chen, ESRI, US
- Nicholas Chrisman, U. of Washington, US
- Gyorgy Fekete, NASA/Goddard, US
- Michael Goodchild, U. of California, US
- Hrvoje Lukatela, Calgary, Alberta, CN
- Hanan Samet, U. of Maryland, US
- Denis White, Oregon State U., US

Some of the questions that these panelists might address include:

- How can HT help spatial database management and analysis?
- What are "Tessellar Arithmetics" and how can they help?
- How does HT compare to raster and vector data models?
- How do properties of triangles compare with squares'?
- What properties do HT numbering schemes have?
- How does HT handle data accuracy and precision?
- Are there optimal polyhedral manifolds for HT?
- Can HT be used to model time as well as space?
- Is Orthographic the universal HT projection?

Panelists were encouraged to provide abstracts of position papers for publication in the proceedings. Those received are reproduced below.

Combination of Global and Local Search strategy in Regular Decomposition Data Structure by Using Hierarchical Tessellation

Zi-Tan Chen, Ph.D
Environmental Systems Research Institute
380 New York St., Redlands, CA 92373, USA
Phone: (714) 793-2853
email: zitan@esri.com

This presentation describes a role of hierarchical tessellation (HT) in a very large spatial data base. Large amount of spatial data can be indexed by regular decomposition data structures, like Quad Trees (QT), Quaternary Triangular Mesh (QTM), etc. All regular decomposition data structures have advantages of simple computation and elegant hierarchy, and facilitate global search.

On the other hand, most spatial features in the real world are irregular in shape and size. Therefore, an irregular decomposition data structure, for example a TIN, has more efficiency and impact in terrain surface feature representation. An irregular decomposition data structure usually has its own local neighbor finding properties. These properties can provide valuable fast local search in special cases. For instance, TIN has its own properties that make it easy to find a neighbor triangle from any given triangle. However, it is not easy to build a global search for a TIN, because its irregular shapes causes difficulties in building a hierarchy.

An optimal search strategy is a combination of the global and the local search in a large spatial data base environment. In this way, a search can benefit from both global hierarchical search and local neighbor finding properties.

The HT explores the local properties of a regular decomposition data structure. Based on knowledge of the HT, fast local search in regular data structure for features with irregular shape becomes possible. This paper discusses the concept. As an example, some experimental results of quadtree indexes a TIN for search triangles are given.

Rendering and managing spherical data with Sphere Quadtrees

Gyorgy Fekete
SAR at National Space Science Center
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
email: gyuri@ncgl.gsfc.nasa.gov

Most databases for spherically distributed data are not structured in a manner consistent with their geometry. as a result, such databases possess undesirable artifacts, including the introduction of "tears" in the data when they are mapped onto a flat file system. Furthermore, it is difficult to make queries about the topological relationship among the data components without performing real arithmetic. The sphere quadtree (SQT), which is based on the recursive subdivision of spherical triangles obtained by projecting the faces of an icosahedron onto a sphere, eliminates some of these problems. The SQT allows the representation of data at multiple levels and arbitrary resolution. Efficient search strategies can easily be implemented for the selection of data to be rendered or analyzed by a specific technique. Furthermore, sphere quadtrees offer significant potential for improving the accuracy and efficiency of spherical surface rendering algorithms as well as for spatial data management and geographic information systems. Most importantly, geometric and topological consistency with the data is maintained.

Implementing a Global GIS Using Hierarchical Tessellations

Michael F. Goodchild
National Center for Geographic Information and Analysis
University of California, Santa Barbara CA 93106
email: good@topdog.ucsb.edu

The QTM scheme described by Dutton and based on recursive subdivision of the faces of an octahedron provides a convenient and practical way of representing distributions over the surface of the earth in hierarchical, tesselated fashion. This presentation describes work at NCGIA Santa Barbara to implement a global GIS using a version of Dutton's scheme. The system makes use of the 3D display capabilities of a graphics workstation. Algorithms have been developed for the equivalent of raster/vector conversion, including filling, and the representation of lines in chain codes. Windowing is simple because of the basic transformations used to create the scheme, suggesting its use in tiling global databases. Examples are given of displays using the system, and of some simple forms of analysis.

The Truncated Icosahedron as the basis for a global sampling design for environmental monitoring

A. Jon Kimerling
Department of Geosciences
Oregon State University
Corvallis, OR 97331

Denis White
NSI Technology Services Corp.
US EPA Environmental Research Laboratory
200 SW 35th St.
Corvallis, OR 97333

A comprehensive environmental monitoring program based on a sound statistical design is necessary to provide estimates of the status of, and trends in, the condition of ecological resources. A systematic sampling grid can provide the adaptive capability required in a broad purpose monitoring program, but how shall the globe or large areas of it be covered by such a grid? Criteria for determining the cartography and geometry of the sampling grid include equal areas across the domain of sampling, regular and compact shape of sampling areas, and hierarchical enhancement and reduction of the grid.

Analysis of systematic subdivisions of projections of the Platonic solids (tetrahedron, hexahedron, octahedron, dodecahedron, and icosahedron) onto the globe show that subdivisions of the dodecahedron and icosahedron produced the most regular set of triangles, but differences among triangles are unacceptably large. In addition, analysis of Lambert azimuthal equal-area map projections for the triangular subdivision of each Platonic solid show that distortions in shape reach unacceptably large maximum values for each solid.

Acceptably small shape distortions (maximum about 2%) can be obtained by subdividing the globe into a truncated icosahedron, an Archimedean polyhedron (commonly used as the tessellation for soccer balls) consisting of twenty hexagons and twelve pentagons. A hexagon face of the truncated icosahedron can be positioned

to cover the entire conterminous U.S.; adjacent hexagons cover Alaska and Hawaii. A hexagon from this model can be decomposed into a grid of equilateral, equal-area triangles whose vertex positions can be projected into geodetic latitude and longitude coordinates on the spheroid.

This triangular grid of sample points has advantages in spatial analysis over square or hexagonal grids. The geometry for enhancement and reduction provides for a hierarchical structure, and includes provision for density changes by factors of 3, 4, and 7. Points in the triangular grid placed on the truncated icosahedron hexagon can be addressed with a hierarchical system based on the systematic decomposition from the hexagon, or by a system similar to the quadrangle labeling convention of the U.S. Geological Survey.

The Evolution of Geopositioning Paradigms

Hrvoje Lukatela
2320 Uxbridge Drive
Calgary, Alberta, CANADA - T2N 3Z6
email: lukatela@uncamult.bitnet

Numerical geopositioning paradigms, used in virtually all of the current geometronical systems, are derived by a simple-minded transplant of the procedures and numerical apparatus of the classical cartography into the realm of digital computing. Such systems suffer from two major faults:

1) Full functionality of their spatial modeling is restricted to a single planar area; usually less than one percent of the planetary surface. Modeling of the time-space relationships between the near-space objects and the terrestrial surface is imprecise and/or inefficient.
2) Geometry relationship derived from the planar digital model is, at best, an imprecise approximation of the corresponding relationship in the object space; at worst, it is an exact opposite.

The purpose for the creation of most geometronical systems is no longer the automated production of an analog, graphical model, from which a human observer derives spatial relationships. In many disciplines, the human map user has been partially or completely replaced by a layer of discipline-specific software; layer which depends on the geometronical system - and its database - for the selection, manipulation and delivery of digitally encoded spatial information. Among the spatial processing requirements of a typical application system, two stand out:

1) From the numerical representation of spatial objects, their geometric relationships - unions, intersections, proximity sections, distances, etc. - must be derived, with at least an order of magnitude higher spatial resolution and precision than that which is employed by the measurements and activities carried out in the object-space.
2) Among a large number of objects, populating the digital model and its database, the system must select those that conform to a criterion based on the spatial extent of another, possibly transient, object. This criterion is frequently combined with non-spatial selection criteria.

HIPPARCHUS is one among a number of new numerical geopositioning paradigms, which provide these facilities, while avoiding the faults mentioned above. Its reference surface - and the data domain - is an ellipsoid of rotation; its repertoire of spatial primitives consists of points, lines and areas on the terrestrial surface, as well as the time-position relationship of the closed orbits and their sensor geometry.

In order to construct an index into a collection of objects, the domain is partitioned, and the fragment identifiers are used as the search arguments of either an implicit or explicit index table. Spherical or spheroidal surface partitioning presents a greater challenge than that of a planar one; beyond five Platonic solids, spherical surface can not be sub-divided into a finite number of regular polygons. The spherical surface can be partitioned using one of the two divergent structure classes: pseudo-regular (Pythagorean) or irregular (Platonic). Of the latter, spheroidal equivalent of the planar Voronoi tessellation - combined with the vector algebra based manipulation of the spherical coordinates - seems to yield an extremely efficient implementation of the critical spatial algorithms. It is, however, the combination of both partitioning techniques that will likely provide a base for the future numerical geopositioning paradigms.

VISUALIZING THE QUALITY OF SPATIAL INFORMATION

Barbara P. Buttenfield
NCGIA, Department of Geography, SUNY-Buffalo, Buffalo, NY 14261
email GEOBABS@UBVMS

M. Kate Beard
NCGIA, Department of Surveying Engineering
University of Maine, Orono, Maine 04469
email BEARD@MECAN1

DESCRIPTION AND SCOPE OF THE ISSUE

Technology currently allows us to process and display large volumes of information very quickly. Effective use of this information for analysis and decision making presupposes that the information is correct or reasonably reliable. Information on the quality of data is essential for effective use of GIS data: it affects the fitness of use of data for a particular application, the credibility of data representation and interpretation, and the evaluation of decision alternatives. The credibility of spatial decision support using GIS may indeed depend on the incorporation of quality information within the database and the display. As Goodchild (1990) states the best insurance will be to sensitize the GIS user community to accuracy issues and to develop tools which allow spatial data handling systems to be sensitive to error propagation. Visualization should be explored as a method for capturing, interpreting and communicating quality information to users of GIS. Clearly, the quality of information varies spatially, and visual tools for display of data quality will improve and facilitate use of GIS. At present, those tools are either unavailable (in existing GIS packages) or not-well developed (error models and the process of visualization are only recently beginning to be addressed directly as research topics).

The quality of spatial data and databases is a major concern for developers and users of GIS (Chrisman, 1983). The quality of spatial information products is multidimensional, and relates to accuracy, error, consistency and reliability. Implications for spatial analysis and for spatial decision-making are too complex for a comprehensive inventory, but can be identified in theoretical work (for example in spatial statistics) as well as in GIS applications (for example in resource management). This paper presents an initial framework for discussion of the role of visualization for understanding and analyzing information about the quality of GIS data. The discussion will proceed from and expand upon the ideas presented here in a panel session at the meeting.

This paper represents part of Research Initiative #7, "Visualizing the Quality of Spatial Information", of the National Center for Geographic Information and Analysis, supported by a grant from the National Science Foundation (SES-88-10917); support by NSF is gratefully acknowledged.

Our goal in this research panel is to bring together representatives from academia, federal agencies and the private sector to present their needs for knowledge about the quality of spatial data products. Discussion will focus on effective means to manage and visually communicate components of data quality to researchers, decision-makers and users of spatial information, particularly in the context of GIS. The intention is to consider a variety of perspectives on topics for a research agenda available to the general GIS community, and to hear the various sectors (educational, commercial and applications) express priorities for topics in the agenda.

THEMES FOR RESEARCH

Questions and impediments relating to the visualization of data quality conceivably cover a very broad ground. For example, issues of modeling and sensitivity analysis might be considered to determine what visual tools are appropriate for particular models, the opportunity for visualization to facilitate spatial analysis, and caveats to consider in implementing visual tools in modeling. The role of visualization in geographical analysis and its role in hypothesis testing and data exploration have been recently reviewed (Buttenfield and Mackaness, 1991), but these topics lie beyond a manageable scope for the panel. Instead, impediments and research priorities within four categories will be addressed. These include defining components of data quality, identifying impediments for maintenance of data models and databases, addressing representational issues, and evaluating particular user needs for data quality information.

Data Quality Components. Perhaps the most commonly cited component of data quality relates to measures of error. Commonly recognized errors include those associated with data collection(source error) and the processing of data for map compilation (process error). Information on source error is often discarded with the completion of map compilation. Process errors have proven difficult to analyze in many cases, for example in studies of digitizing error, or in modeling error associated with soil mapping (Fisher, 1991). In statistics, the concept of Least Squares Error has been applied to determine reliability (or what is called 'confidence') in hypothesis testing. A third error component (use error) is associated with the appropriate application of data or data products (Beard, 1989).

By some definitions, error (the discrepancy between measurement and true value) is much more difficult to assess than accuracy (the discrepancy between measurement and a model). The best examples of this may be found in determination of geodetic position, which until the development of GPS systems was limited to (albeit precise) projection of location with reference to a geodetic spheroid and datum. The Proposed Standard for Digital Cartographic Data Quality (Moellering, 1988) incorporates three accuracy measures (positional accuracy, attribute accuracy, and consistency) in addition to lineage and completeness.

A standard definition of data quality and its components may be difficult to agree upon, as the domain of an application will likely impact the user needs. For soils data, for example, requirements for consistent attribution of soil type are more readily evaluated than requirements for accurate positioning of soil parcel boundaries. For demographic data, where enumeration boundaries are determined by mechanisms unrelated to the particular variable at hand, just the converse may be true. Regardless, there should be consensus about some of the research priorities for this theme:

> What visual tools are appropriate for particular error models?
>
> How can visualization facilitate monitoring of error propagation?

<u>Data Models and Database Issues</u>. Management of data quality within a GIS database requires attention during manipulation and update, and will likely impact upon the future architecture of such databases for implementation. Information about the information within a database is referred to as metadata, and has recently become a research issue in its own right (see for example Lanter and Veregin, 1990). The representation of data quality components in a data structure will not only have requirements to facilitate their visual display, but also must be implemented with efficient pointers and links to facilitate update operations. Analysis of error propagation might also be facilitated by visual display, and the design of these graphic tools may not be closely aligned with the design of conventional GIS graphics. This will be covered under the third theme presented below. Other questions arise:

> How can the metadata be updated simultaneously with the data?
>
> What database requirements must be implemented to accommodate real-time data quality representations for static GIS products, or for dynamic displays?
>
> Can current data structuring alternatives accommodate changes to data and data quality in effective ways? How can links b/t data and data quality be preserved during database modification or update?

<u>Representational Issues</u>. The ease with which visualization tools may be integrated within GIS packages varies considerably depending on at least three issues, including the domain of the phenomena to be studied, the purpose or intent of the user, and the format of the GIS software (MacEachren, Buttenfield, Campbell, and Monmonier, 1991). This presents a substantial challenge to the system designer. Buttenfield and Ganter (1990) suggest that GIS requirements for visualization include conceptual, technological, and evaluatory solutions, which may be seen to vary over three broad domains: inference, illustration, and decision-making. Each presents a challenge to the integration of appropriate visualization tools.

Maps are a major tool for decision-making with GIS. Current GIS software includes functions to create cartographic output automatically or interactively. However, none of the current turnkey systems include mechanisms to ensure correct use of graphics functions. Poorly designed maps may convey false ideas about the facts represented by the data, and bias the decision-making process. Weibel and Buttenfield (1988) explore ways to improve the quality of GIS map products, and increase effectiveness of information transfer based on graphics. Their guidelines may provide only a rudimentary implementation for visualizing data quality. Research priorities that come to mind under this theme may involve both system benchmarking and cognitive evaluations, as seen for example by the following questions:

> What design tools are appropriate for graphical depiction of data quality?
>
> Will generation of realtime data displays during database update facilitate monitoring of error and error propagation?
>
> How can the effectiveness of such displays be evaluated? For example, What is the utility of embedding data quality with data in graphic display? Can the two be merged, or is this too much of a cognitive challenge for effective interpretation?

Evaluation of User Needs. Ganter (1990) discusses visualization from a cognitive as opposed to graphical perspective, cautioning readers that discovery and innovation, which have traditionally involved thinking visually and producing images, increasingly benefit from GIS and CAD. He argues for the importance of understanding the human faculties which use pictures as tools in thinking. Science and engineering define problems, explain processes, and design solutions through observation, imagination and logic. Evaluation of user demands for data quality information will require sensitivity to the internal (perceptual and cognitive) mechanisms by which spatial and temporal patterns are interpreted.

Equally important is the need for sensitivity to the domain of the GIS application. For example, reliability associated with a routing of emergency dispatch vehicles will likely vary with each link of the route; this information must be presented with high precision and in a short timeframe. Reliability variations associated with the environmental impact of a timber clear-cut operation cannot be tied to a routed network, and variations may be interpolated as opposed to tabulated raw data. In this context, some research questions may be proposed:

> What are expectations of GIS users regarding data quality displays?
>
> How will visualization of data quality impact upon the reliability and credibility of spatial decision-making using GIS?

SUMMARY

With advances in technology, storage and displays mechanisms are now in place for real-time display not only of spatial pattern but also of the quality of the rendered data. Developments in software provide spatial inference and statistical explanation to the verge of providing models about the reliability and consistency of spatial interpretation, and this has paved the way for application of GIS to policy-making and decision support. There is a need and timeliness to consider data quality issues in the context of GIS. Our acuity for visual processing indicates that current technology in graphical display may assist our efforts to validate the decisions and results based on GIS analyses. The panel discussion is intended to present multiple viewpoints and to encourage the research and user community to address visualization of data quality as an attainable goal in the development of GIS.

REFERENCES CITED

Beard M K (1989) Use error: The neglected error component. **Proceedings,** AUTO-CARTO 9, Baltimore, Maryland, March 1989: 808-817.

Buttenfield B P and Ganter J H (1990) Visualization and GIS: what should we see? what might we miss? **Proceedings,** 4th International Symposium on Spatial Data Handling, Zurich Switzerland, July 1990, vol.1: 307-316.

Buttenfield B P and Mackaness WA (1991) Visualisation. Chapter II.a.4 in Maguire, D. Goodchild, M.F. and Rhind, D (Eds.) **GIS: Principles and Applications**. London: Longman Publishers Ltd. (in press)

Chrisman N R (1983) The role of quality information in the longterm functioning of a geographic information system. **Cartographica** 21(2): 79-87.

Fisher P F (1991) Modeling Soil Map-Unit Inclusions by Monte Carlo Simulation. **International Journal of GIS** (in press).

Ganter J H (1988) Interactive graphics: linking the human to the model. **Proceedings,** GIS/LIS '88, pp. 230-239.

Goodchild, M F (1990) Spatial Information Science. **Proceedings,** 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, July 1990, vol.1: 3-12.

Lanter D P and Veregin H (1990) A lineage meta-database program for propagating error in geographic information systems. **Proceedings** GIS/LIS '90, Anaheim, California, November 1990, vol.1, 144-153.

MacEachren A E, Buttenfield B P, Campbell J C and Monmonier M S (1991) Visualization. In Abler, R. A., Olson, J. M. and Marcus, N. G. (Eds.) **Geography's Inner World**. Washington, D. C.: AAG (forthcoming).

Moellering, H (1988) The proposed standard for digital cartographic data: report of the digital cartographic data standards task force. **The American Cartographer,** 15(1) (entire issue).

Weibel, W R and Buttenfield B P (1988) Map design for geographic information systems. **Proceedings,** GIS/LIS 88, November 1988, San Antonio, Texas vol.1: 350-359.

A General Technique for Creating SIMD Algorithms on Parallel Pointer-Based Quadtrees

Thor Bestul
Center for Automation Research
University of Maryland, College Park, MD 20742

Abstract

This paper presents a general technique for creating SIMD parallel algorithms on pointer-based quadtrees. It is useful for creating parallel quadtree algorithms which run in time proportional to the height of the quadtrees involved but which are independent of the number of objects (regions, points, segments, etc.) which the quadtrees represent, as well as the total number of nodes. The technique makes use of a dynamic relationship between processors and the elements of the space domain and object domain being processed.

1 Introduction

A quadtree is a data structure for indexing planar data. It is a tree with internal nodes of degree four, where the root represents a planar rectangular region, and the four sons of each internal node represent the four quadrants of the node's region. Generally, each node stores some information about the region it represents and also a color, with the internal nodes being considered gray and the leaf nodes having some color derived from the data in their regions. A particular variety of quadtree is typically defined by giving a *decomposition rule*, which determines whether a region should be subdivided and the corresponding node given sons. For example, for quadtrees to represent binary images (i.e. *region quadtrees*), the rule is that if a region contains pixels with both binary values, it is decomposed and the corresponding node is an internal gray node with four sons. If a region consists entirely of only one the binary values, the corresponding node is a leaf node and has, say, the color black if the single value is 1 and white if it is 0. For our purposes, the division of a region into quadrants is always done uniformly, although the definition of quadtree does not necessitate this.

This paper describes a technique for creating quadtree algorithms intended to run in a parallel processing environment with many processors sharing a single instruction stream (Single Instruction stream Multiple Data stream or SIMD) and possessing a general facility for intercommunication among the processors. The algorithms are for building and processing quadtrees stored with one quadtree node per processor, and with non-leaf node processors possessing pointers to the processors representing their sons. We call such a quadtree implementation a parallel pointer-based quadtree.

The target architecture consists of many (several thousand) processors each with a modest amount (a couple Kbytes) of local storage, and all of which simultaneously perform a single sequence of instructions in lockstep. The exception to this is that each processor can ignore instructions depending on the current values in its local memory. Each processor has access to anywhere within the local memory of any of the other processors. Thus if each processor possesses a pointer to data in some other processor's memory, they may all dereference their pointers in lockstep. Simultaneous reads of data from a single location are supported. Simultaneous writes of data to a single location are also supported, as long as a contention resolution operation is specified along with the write operation, such as summing the received values, min or maxing them, performing various boolean operations on them, or selecting one of them arbitrarily.

A fundamental operation used often in the following is that of processor *allocation*, in which a processor obtains a pointer to some other processor not in use and initializes its local memory in some fashion, causing it to then become part of the active computation underway. Processors can also be *de-allocated*, meaning that they no longer contribute to the active computation, and lay idle waiting to be allocated again by an active processor. This can be done in parallel for many processors which desire to allocate other processors by using the *rendezvous* technique [4].

This algorithm creation technique combines two paradigms for parallel computation in the arena of spatial data structures and the objects they represent. One paradigm is space parallelism, in which the two or three-dimensional space represented by our data structure (in this case a quadtree) is divided up among the processors, each of which operates serially across the entire set of objects. The other paradigm is object parallelism, in which the set of objects involved is divided up among the processors, each of which operates serially across the entire space. The technique described here uses parallelism across both space and the set of objects. In order to accomplish this combination of paradigms the technique leans heavily on the facility of general intercommunication among processors, and in particular on the capability of the handling of multiple reads and writes.

The technique succeeds partially because of its use of a very fine-grained parallelism in which we have parallel processors distributed both across the spatial elements and across the objects in the object set. However, the technique only attains its full generality when we discover a mechanism to press beyond even this level of granularity when necessary, and to make use of a a dynamic relationship between processors and the elements of the space and object domains being processed.

2 A Degenerate Case

It is simplest to describe the algorithm creation technique by first describing a degenerate case of it. We use as an example problem the task of constructing a PR quadtree for a collection of points in a plane, and create a parallel algorithm for this task. In the construction of a PR quadtree, a node should be assigned the color gray and subdivided if more than one point lies within its boundary. A node should be white if it contains no points and black if it contains exactly one.

We allow a processor for each point, containing that point's coordinates, and initially have one processor to represent the root of the quadtree under construction. The algorithm consists of a single loop iterating from the top to bottom level of the quadtree being built, constructing all the quadtree nodes for each particular level at once. Each point processor contains a pointer to a node processor, and initially all the point processors point to the single root node processor. The algorithm is illustrated in Figures 1 through 4 for a small set of points.

For the first iteration of the loop, all points retrieve from the root node information about where the boundary of the region it represents lies, and then compute whether or not they lie within that boundary. All points outside of the root boundaries set their node pointers to null. All points with non-null node pointers then send the value "1" to the root node. These values are summed at the root node as they are received. The root node processor then checks this sum, and if it is greater than 1, meaning that more than one point lies within its boundary, it assigns itself the color gray, and allocates processors for its four son nodes. If the sum is 0 then the node assigns itself the color white, and black if the sum is 1. Each point then checks the node it points to (still the root node during this first iteration), and if it is gray, computes which quadrant of the node it is in, and fetches the corresponding son pointer. Each point processor now possesses a pointer to the node processor corresponding to the quadrant within which the point lies, if it lies in any, or has a null node pointer otherwise.

On the second iteration of the loop each point again sends the value "1" to the node to which it points, the nodes check the sums they receive, and all those nodes found to have more than one point within their boundaries are set to gray and allocate sons. Each point processor then selects the appropriate son to point to. This process is repeated moving down the quadtree being created until no node has more than one point within its boundary, or some limit on the number of quadtree levels has been reached.

Below is the PR-quadtree construction algorithm. The main procedure is 'PR_quadtree()'. This procedure takes as an argument a pointer to a node processor, which it uses as the root of the quadtree constructed. Only those point processors active when the routine is called are used for the construction of the quadtree. This is so that some subset of all the stored points could be selected as a basis for the quadtree, by having the routine called from within a parallel conditional statement. The effect of a conditional statement in a parallel context is to deactivate for the duration of the statement those processors whose currently stored values do not satisfy the conditional, as will be discussed below.

In the procedure 'PR_quadtree', every node is given a flag called *live*. At the beginning of each iteration of the procedure's loop, we set the *live* flag to true in all those nodes which have some point processor pointing to them, and to false in all other nodes. Only those nodes for which *live* is true are operated on during the rest of the iteration.

We give here a description of our programming paradigm and algorithm notation.

The algorithm is given as a procedure, possibly with other supporting procedures. There is no nesting of procedure declarations. Besides procedures there are global variables; a variable is global if it is declared outside of any procedure. A procedure may return a value; if so, the type of the value is given before the 'procedure' keyword.

A variable can be either parallel or non-parallel; in the latter case we say that the variable is mono. Every parallel variable belongs to some particular parallel processor type, such as point processor or node processor. A variable is declared as mono with the construct "mono *declaration*" and as parallel with the construct "parallel *processor-type declaration*". In certain contexts the default type is mono and in certain other contexts the default is parallel; in these contexts the prefixes "mono" or "parallel *processor-type*" may be omitted. Global declarations of variables are mono by default, so when global declarations of parallel variables are made the prefix "parallel *processor-type*" must be used, and furthermore the declarations must be given in a record-style block delimited by the keywords 'begin' and 'end'. Only one such global block of variables is permitted for each processor type. Each processor type has its own namespace for global variables.

If a variable is a pointer, both the pointer itself and the type of object pointed to can be either parallel or mono. The declaration "mono pointer mono integer p" specifies a simple mono pointer to a mono integer, and corresponds to the usual notion of pointers in serial architectures. The declaration "mono pointer parallel apple integer p" specifies that 'p' gives a uniform offset into the storage of all processors of type 'apple' and that at that offset is found a parallel variable of type integer. The declaration "parallel apple pointer parallel orange integer p" specifies that the parallel pointer 'p' belongs to the processors of type 'apple', and that each instance of 'p' points to some datum of integer type in some processor of type 'orange'. The declaration "parallel apple pointer mono integer p" specifies that the parallel pointer 'p' belongs to the processors of type 'apple' and that each instance of 'p' points to some mono integer datum.

Procedures can also be either parallel or mono. A procedure is specified as parallel by prefixing its declaration with "parallel *processor-type*", otherwise it is taken to be mono. The arguments, local variables, and return value of a mono procedure are taken to be mono by default. The arguments, local variables, and return value, if any, of a parallel procedure are by default parallel values. Furthermore, within the body of a parallel procedure, parallel global names are interpreted in the context of the processor type given in the procedure declaration. Any of the parameters, local variables, or return values of any procedure can be forced to be of some type other than their default by use the 'mono' and 'parallel' prefixes.

Declarations of procedure parameter types are given between the procedure argument list and the procedure's 'begin' statement.

The construct "in_every *processor-type* do *statement-list*" causes precisely the set of all processors of the given type to become active at the beginning of the statement list. Furthermore, within the statement list the names of global variables are interpreted in the context of the given processor type. At the end of the statement list, the set of processors which were active before the statement list was entered is re-established as the active set.

The are two variants to "in_every". One is "in_every *boolean-expression* *processor-type* do *statement-list*" and the other is "in_every *processor-type* having *boolean-expression* do *statement-list*". Both are equivalent to "in_every *processor-type* do if *boolean-expression* then *statement-list*".

Outside of a parallel procedure for a given type or an 'in_every' statement for a given processor type, use of parallel global variable names for that processor's type is considered an error.

In the construct "if *conditional* then *statement-list*", if the conditional is a parallel expression (one involving parallel variables or values), then the subset of the present active set of processors whose current values satisfy the conditional are made the active set at the beginning of the statement list. At the conclusion of the statement list, the set of processors which were originally active is re-established as the active set.

When a parallel procedure is called, the set of processors active at the time of invocation will be the set active at the beginning of the execution of the procedure body.

If 'p' is a pointer to a processor, and 'f' is a parallel variable in the processors of the type of that pointed to by 'p', then the notation 'f<p>' indicates the value of the variable 'f' in the particular processor pointed to by 'p'.

The symbol '<–+' indicates an assignment statement involving a possibly-multiple write, and in which the write contention is to be resolved by summing the multiply-written values. There are other similar assignment symbols such as '<–or' and '<–min'.

The algorithm is as follows:

```
node || pointer node father;
node || pointer node array son[4];
node || integer level;
node || node_color color;
node || real left, right, bottom, top;

point ||  real x, y;


node || procedure allocate_sons();
/* Allocates four sons for each node active when
   the procedure is called and fills each node's
   son array accordingly. */


procedure PR_quadtree(root)
pointer node root;
/* Builds a PR quadtree for all the points which are in
   allocated point processors. Assumes that root points to
   a node whose level and boundary have been initialized,
   and uses this node as the root of the quadtree
   constructed. */
begin
  node || integer total;
  point || pointer node node_ptr;
  point || integer my_quadrant;
  integer l;

  /* Make all points within the root's boundary point to the
```

```
     root, give all other points NULL node pointers. */
  in_every point do
    begin
      node_ptr <- NULL;
      if x >= left<root> and x <= right<root> and
         y >= bottom<root> and y <= top<root> then
        node_ptr <- root;
    end;

  /* Loop from level of root to bottom. */

  for l <- level<root> downto 0 do
  begin
    /* Initialize point total for all nodes on present
       level to zero. */
    in_every node having level = l do
      total <- 0;

    /* Each point contributes 1 to the point total for the
       node containing it. */
    in_every point having node_ptr <> NULL do
      total<node_ptr> <-+ 1;

    /* Nodes with no points in them are white.  Nodes with
       one point in them are black.  Nodes with more than
       one point in them are gray.  If we're not at the bottom
       level we allocate sons for the gray nodes. */
    in_every node having level = l do
      begin
        if total = 0 then color <- WHITE
        else if total = 1 then color <- BLACK
        else begin
          color <- GRAY;
          if l > 0 then allocate_sons();
        end;
      end;

    /* If at bottom level then we're done. */
    if l = 0 then return;

    /* The points in each gray node divide themselves among
       the sons. */
    in_every point having node_ptr <> NULL and
                          color<node_ptr> = GRAY do
      begin
        /* Each point determines which subquadrant it is in. */
        my_quadrant <- 0;
        if x > 0.5 * (left<node_ptr> + right<node_ptr>) then
          my_quadrant <- my_quadrant + 1;
        if y < 0.5 * (bottom<node_ptr> + top<node_ptr>) then
          my_quadrant <- my_quadrant + 2;

        /* Each point fetches the pointer to the corresponding
           node son. */
        node_ptr <- son<node_ptr>[my_quadrant];
      end;

  end;
end;
```

To summarize the technique so far, we allow one processor per object and one processor per quadtree node. Each object is given access to a sequence of shrinking nodes which contain it; initially all objects have access to the root node. By having each object obtain information from its node, and by combining at the node information from all of the objects who access that node, the objects make decisions about descending the quadtree from that node.

3 The General Technique

Now consider instead the task of constructing a PM quadtree for line segment data. In constructing a PM quadtree, a node should be assigned the color gray and subdivided if its boundary contains more than one endpoint, or if its boundary has two segments which enter it but which do not have a common endpoint within it. Initially, we have one processor allocated for the quadtree root, and one processor for each line segment, containing the coordinates of the segment's endpoints.

Consider creating an algorithm, similar to the one given above, to construct the PM quadtree for this segment data. Each segment processor initially possesses a pointer to the quadtree root processor. Each segment processor computes how many of its segment's endpoints lie within the boundary of the node to which the segment processor points; this will be 0, 1, or 2. Each segment then sends this value to the node it points to, and both the maximum and minimum of these values are computed at the node. Any node which receives a maximum value of 2 assigns itself the color gray, since this means that some single segment has both endpoints in the node's boundary. Any node which receives a maximum of 1 and a minimum of 0 also assigns itself the color gray, since this means that there are at least two segments in the node's boundary, one which passes completely through it and one which terminates within it.

Then each segment with exactly 1 endpoint in the node it points to sends the coordinates of that endpoint to the node. The node receives the minimal bounding box of the coordinates sent to it (this, of course, amounts simply to applying min and max operations appropriately to the coordinate components). If this minimal bounding box is larger than a point, the node assigns itself the color gray, since this means that some two segments entering the node have non-coincidental endpoints within the node.

Finally each segment with 0 endpoints in the node it points to determines whether it in fact passes through the interior of the node at all, and if so it sends the value "1" to the node, where these values are summed. If the sum received by the node is greater than 1, the node assigns itself the color gray, since this means that some two segments passing through the node do not have any endpoints in the node, which implies that they do not have a common endpoint in the node. Then all gray nodes allocate son processors. Any nodes which were not given the color gray should be colored white if no segments entered their interior (the sum is zero), and black otherwise (the sum is one).

At this point in the algorithm, we would like to have all segment processors which point to gray nodes compute which of the node's sons they belong to, and retrieve from the node the appropriate son pointer, just as in the case of the PR quadtree construction algorithm. Of course in this case, as opposed to the case of the point data, a given segment can intersect more than one of the node's sons, and we are left with the situation of wanting to assign up to four son pointers to the segment processor's node pointer, and processing each of the corresponding sons. The solution to this dilemma is to allocate *clones* of each such segment processor, that is, to create multiple processors which represent the same segment, and all of which contain (almost) the same information. So for each segment processor pointing to a gray node, we allocate three clone processors, all of which contain the segment's endpoints and a pointer to the same node as the original segment processor. In addition, the original and its clones each contain a *clone index* from 0 to 3, with the original containing 0 and each of the clones containing a distinct index from 1 to 3. Now the original and its clones each fetch a son pointer from the node that they all point to, each one fetching according to its clone index, so that each gets a different son pointer.

The subsequent iterations of the algorithm proceed as the first, with each segment processor determining how many of its endpoints lie within the interior of the node it points to, and with the eventual computation of the colors of all the nodes on each particular level. At this point in each iteration, notice that any segment processors pointing to leaf nodes, or whose segments do not pass at all through the interior of the node to which they point, will not have any further effect of those nodes, and can thus be de-allocated and re-used later. This reclaiming of segment processors keeps the number of clones allocated for each segment from growing exponentially. In fact the number of processors required for a given segment at a given level in the construction of the quadtree will be only roughly as many as there are nodes in that level of the tree through whose interior the segment passes.

To summarize the general technique then, we allow one processor per quadtree node, and initially allow one processor per object. Each object is given access to a sequence of shrinking nodes which contain part of it; initially all objects have access to the root node. By having each object obtain information from its node, and by combining at the node information from all of the objects who access that node, the objects make decisions about descending the quadtree from that node. For those objects which do descend, it is desirable for their various parts which lie in various quadrants of the node to descend in parallel. Thus we allow duplicate or 'clone' processors for each object, and have each processor handle just that portion of the object relevant to one quadrant of the node. Duplicate processors which determine that they can no longer effect the the node to which they point, because that node is a leaf, or because the object they represent does not overlap that node, can deactivate themselves so that they may be used later in the computations for some other object.

We see then that this technique allows us to go beyond the level of granularity of one processor for every element (space component or object) to a level where there are multiple processors for certain elements and none for others; where the processors are being used and disposed in a dynamic fashion.

4 Other Applications

The same general technique can be applied to create algorithms for several other quadtree tasks. For example consider the task of shifting a quadtree. Suppose we have already created somehow a quadtree with one processor per node, and wish to compute a new quadtree to represent the original one shifted by some amount. Using this technique we create the following algorithm.

Have each black leaf node of the old quadtree compute its own shifted position. Then allocate a new processor for the root node of the new (shifted) quadtree, and give each old black leaf node a pointer to this new root node. Iterate the following from top to bottom of the new quadtree.

Each old black leaf node fetches the boundary of the new node it points to, and computes whether, in its new shifted position, it encloses that node. All old black leaf nodes which do enclose the new node they point to send the value TRUE to the new node, which combines the received results by or-ing them. Any new node which thus determines it is enclosed by some old black leaf node assigns itself the color black. Then each old black node computes whether it intersects the new node it points to even if it doesn't enclose it, and if so sends TRUE to the new node, which combines the received results by or-ing them. The new node then assigns itself the color gray if it is not already black and if some old black leaf node intersects it, i.e. if the received result is TRUE. Any new node which does not determine itself to be black or gray in this way assigns itself the color white. All new gray nodes allocate sons for themselves. Each old black leaf node pointing to a new gray node allocates clones for itself, and divides up among itself and its clones the son pointers of the new gray node to which they all point.

In the above procedure, before clones are allocated, any processor representing an old black leaf node which points to a black or white new node should de-allocate itself so that it may be re-used, since it will no longer affect the new node it points to. Of course, this de-allocation should not be done for those processors which originally represented the quadtree to be shifted, if it is desired that this original quadtree not be lost, but these processors can be specially marked to avoid their being de-allocated.

It is not hard to see how this same technique can also be used to create algorithms for quadtree rotation and expansion which run in time proportional to the height of the new quadtree, by computing in parallel the rotated or expanded version of each old black leaf node, and building the new quadtree using cloning. One can also create algorithms for the simultaneous insertion of many polygons or arbitrary regions into a quadtree. Some of these algorithms will require an additional post-processing phase in which any node with four sons of the same color is given that color and has its sons discarded. This can be done in a single bottom-up pass over the new quadtree in time proportional to its height.

5 A Hidden Edge Algorithm Using Cloning

To show the flexibility of our technique, we use it here to create an algorithm for computing hidden edges in a scene consisting of polygons lodged in 3-

space. The algorithm builds an MX quadtree of the pixels in the viewplane. In an MX quadtree [3], all pixel sized leaf nodes through which an edge passes are black, and all other leaf nodes are white.

This algorithm is based on the Warnock algorithm for hidden edge computation [1] [5]. The essential idea of the algorithm is that while recursively decomposing the viewplane into quadrants, if it can be determined that all of the pixels which compose some entire quadrant at some level of decomposition should be white, then the quadrant does not need to be further decomposed. In order to determine if this is so for a given quadrant, we consider the planes (in 3-space) in which our polygons lie. After computing the projections onto the viewplane of all polygons (which is done in parallel by the polygon processors), we consider the planes of those polygons whose projections completely enclose the given quadrant. We wish to determine if the plane of any of those polygons is "closer" to the viewpoint than the planes of the other polygons whose projections enclose the quadrant. To determine this, we compute the inverse projections of the quadrant corners onto the planes of the enclosing polygons, and if one plane is found to be nearer to the viewpoint for all four corners, it is deemed the closest plane.

The algorithm proceeds as follows. We initially assign one processor per polygon, and have one processor representing the root node of the viewplane quadtree being constructed. Initially each polygon processor possesses a pointer to the root node. The following procedure is iterated from top to bottom of the quadtree being built.

Each polygon computes its projection onto the viewplane (these can be pre-computed since they are fixed), and determines the relationship of its projection with the quadtree node to which it points. Specifically, it determines whether its projection encloses the quadrant, or is involved with it, meaning it overlaps but does not enclose the quadrant, or whether it is outside the quadrant altogether.

Each polygon whose projection encloses its quadrant computes the inverse projection of each of the four corners of its quadrant onto its plane. This computation produces for each corner a distance from the viewpoint to the polygon's plane. Each of these polygons then sends this distance for each of the four corners to its quadrant (node) processor, which computes the minimum of these values as they are received. Each polygon then reads back the minimum distance for each of the four corners, and if all four minimum distances are equal to the corresponding distances which the polygon computed for its own plane, the polygon concludes that its plane is closest to the viewpoint. The polygon then informs its quadrant that it is enclosed by the projection of a polygon whose plane is closest to the viewpoint, and based on this the quadrant assigns itself the color white.

Then all polygons which are involved with (i.e. overlapping but not enclosing) their quadrant send the value TRUE to their quadrant, which combines the values sent to it by or-ing them. Any quadrant not already assigned the color white and which determines it has some polygon involved with it assigns itself the color gray. All other quadrants have no polygons whose projections either enclose them or are involved with them, so they assign themselves the color white. All gray quadrants allocate sons.

Those polygons which point to a quadtree leaf node, or which are outside the quadrant to which they point, de-allocate themselves, since they will no longer affect those nodes. All remaining polygon processors point to a gray nodes. Each remaining polygon processor allocates clones, and divides up among itself and its clones the son pointers of its node.

On the last iteration of the algorithm, that is, the pixel-level iteration, the procedure above is modified so that any node which is involved with some polygon assigns itself the color black instead of gray. After this last iteration, the quadtree constructed is an MX quadtree representation of the viewplane of the projection, with hidden edges eliminated.

Below is the hidden-edge algorithm. The main procedure is 'hidden_edge()', which takes as an argument a pointer to a node processor, and uses this as the root of the quadtree constructed. As with 'PR_quadtree()', only those polygon processors active when the routine is called are used for the construction of the hidden-edge image quadtree.

```
node || pointer node father;
node || pointer node array son[4];
node || integer level;
node || node_color color;
node || real left, right, bottom, top;

/* Vertex projections onto viewplane. */
polygon || real array x[NPOINTS], y[NPOINTS];
/* Number of vertices in polygon. */
polygon || int npts;
/* Parameters of polygon plane. */
polygon || real a, b, c;

polygon || real polygon || procedure poly_plane_dist(x, y);
polygon || real x, y;
/* For each active polygon, returns the distance from the
   viewpoint to the polygon plane via the point (x, y) on
   the viewplane. */

polygon || procedure allocate_clones();
/* Allocates four clones for each active polygon.  The
   clones get the clone indices 0, 1, 2, and 3. */

polygon || procedure deallocate_clones();
/* Deallocate all active clones. */

node || procedure allocate_sons();
/* Allocates four sons for each active node. */

polygon || relation
  polygon || procedure find_relation(left, right, bottom, top);
polygon || real left, right, bottom, top;
/* Each active polygon determines the relationship (INVOLVED,
   OUTSIDE, ENCLOSES) of its projection with the rectangle
   defined by the parameters passed. */

procedure hidden_edges(root)
value pointer node root;
/* Builds a parallel quadtree to represent the scene of
   all the polygons.  Performs hidden edge elimination
   based on a projection using the plane of the quadtree
   leaves as viewplane.  The pointer passed is assumed to
   point to a quadtree node whose level and boundaries
```

```
    have been initialized and is used as the
    root of the quadtree constructed. */
begin
  polygon || relation rel;
  polygon || real pleft, pright, pbottom, ptop;
  polygon || pointer node node_ptr;
  polygon || real pll, plr, pul, pur;
  polygon || integer clone_index;
  integer l;

  /* Start off with all polygon clones pointing to the root. */
  in_every polygon do
    node_ptr <- root;

  /* Loop from level of root node to bottom. */

  for l <- level<root> downto 0 do
  begin
    /* Each polygon fetches the boundaries of the node it
       points to and determines its relationship with it. */
    in_every polygon do
      begin
        pleft <- left<node_ptr>;
        pright <- right<node_ptr>;
        pbottom <- bottom<node_ptr>;
        ptop <- top<node_ptr>;
        rel <- find_relation(pleft, pright, pbottom, ptop);
      end;

    /* Each node on the current level initializes the minimum
       distance for its four corners to be infinity. */
    in_every node having level = l do
      begin
        ll <- INFINITY;
        lr <- INFINITY;
        ul <- INFINITY;
        ur <- INFINITY;
      end;

    /* Every polygon processor whose projection is not outside
       its node determines the distance from the viewpoint to
       the polygon's plane for each of the four corners of the
       node.  For each of the four corners, the minimum
       distance, computed over the set of planes of all such
       polygons, is accumulated at the node processors. */
    in_every polygon having (rel <> OUTSIDE) do
      begin
        pul <- poly_plane_dist(pleft, ptop);
        pur <- poly_plane_dist(pright, ptop);
        pll <- poly_plane_dist(pleft, pbottom);
        plr <- poly_plane_dist(pright, pbottom);

        ul<node_ptr> <-min pul;
        ur<node_ptr> <-min pur;
        ll<node_ptr> <-min pll;
        lr<node_ptr> <-min plr;
      end;

    /* Each node on the current level initializes to FALSE
       a flag which indicates that it is enclosed by the
```

```
   projection of the closest polygon, and to TRUE a flag
   which indicates that the projections of all polygons
   are outside it. */
in_every node having level = l do
  begin
    enclosed_by_closest <- FALSE;
    all_outside <- TRUE;
  end;

/* Each polygon whose projection encloses its node
   determines if its plane is closest (among the planes of
   all such polygons) at all four corners of the node.
   The disjunction of these results is accumulated at the
   node processors. */
in_every polygon having (rel = ENCLOSES and
                         pul = ul<node_ptr> and
                         pur = ur<node_ptr> and
                         pll = ll<node_ptr> and
                         plr = lr<node_ptr>) do
    enclosed_by_closest<node_ptr> <-or TRUE;

/* Each polygon knows if it is outside the node it
   points to.  The conjunction of these results is
   accumulated at the node processors. */
in_every polygon having (rel <> OUTSIDE) do
    all_outside<node_ptr> <-and FALSE;

/* Finally we determine the color for each node on the
   current level. */
in_every node having level = l do
  begin
    if enclosed_by_closest or all_outside then
      color <- WHITE;
    else begin
      if l = 0 then color <- BLACK;
      else color <- GRAY;
    end;
  end;

/* Each polygon clone pointing to a black or white node,
   or which is outside of the node it points to,
   is de-allocated. */
in_every polygon having (color<node_ptr> = WHITE or
                         color<node_ptr> = BLACK or
                         rel = OUTSIDE) do
  begin
    deallocate_clones();
  end;

/* If at the bottom level then we're done. */
if l = 0 then return;

/* Each gray node on the current level allocates sons. */
in_every node having level = l and color = GRAY do
  allocate_sons();

/* Each remaining polygon allocates four clones and
   tags itself as an old clone. */
in_every polygon do
begin
```

```
      allocate_clones();
      in_every polygon do new_clone <- TRUE;
      new_clone <- FALSE;
    end;

    /* Then the new polygon processors each get a pointer
       to one of the node's sons, and the old polygon
       processors are deallocated. */
    in_every polygon do
      if new_clone then
        node_ptr <- son<node_ptr>[clone_index];
      else
        deallocate_clones();

  end;
end;
```

6 Some Timing Results

In this section we present some timing results for the PR quadtree building algorithm and the hidden edge algorithm for implementations of these algorithms on a Connection Machine. A Connection Machine is a SIMD architecture based on a multi-dimensional cube. The vertices of the cube correspond to processors, and the edges correspond to direct communication links between the processors. The illusion of direct access from one processor to the memory of any other is supported by a sophisticated routing algorithm, which deals with bottlenecks and which also supports simultaneous read access and simultaneous write access using several contention resolution operations. Due to the nature of the contention resolution mechanism, the amount of time required to perform a simultaneous write to or read from a single location tends to be proportional to the log of the number of processors performing the simultaneous access. The Connection Machine also support virtual processors, meaning that each processor can emulate several processors, with a proportional reduction in processing speed and memory per processor. The mechanism of virtual processors in transparent to the code which runs on the Connection Machine.

The algorithms were implemented in C*, a parallel version of C, using floating point for all geometric coordinates and were run on a 16384 processor CM-2 without floating point hardware. For each algorithm and number of objects processed, two times are given. One is the real elapsed time, and one is the amount of time spent actually performing operations on the Connection Machine. The tables reveal that the running times of the algorithms on a Connection Machine are not in fact completely independent of the number of objects represented, which was expected since the execution of multiple reads and writes takes time proportional to the log of the number of processors involved in the simultaneous access. This fact, together with the fact that such intercommunication operations tend to be the most time consuming operations on a Connection Machine, explains the approximate log dependency seen in the tables of the algorithm running times on the number of objects represented.

Table 1 shows timing results for the PR quadtree building algorithm for various numbers of points distributed randomly over a square region, for a quadtree with a maximum depth of eight levels.

Number of Points	Elapsed Time (s)	CM Time (s)
10	0.88	0.61
100	2.15	1.78
1000	4.09	3.25
10000	6.98	6.02

Table 1:

Table 2 shows timing results for the hidden edge algorithm for various numbers of square polygons distributed randomly over a parallelpiped region with a pre-computed parallel projection onto a viewplane parallel to one of the faces of the parallelpiped. The MX quadtree constructed has a maximum depth of eight levels, i.e. it is the MX quadtree for a 128 by 128 pixel image.

Number of Polygons	Elapsed Time (s)	CM Time (s)
5	9.49	8.57
50	12.64	11.24
500	18.79	15.49

Table 2:

7 Summary

This paper has presented a technique for creating SIMD algorithms for parallel pointer-based quadtrees. It combines parallelism both across the elements of the space represented by the quadtree and across the elements of the set of objects represented. It produces algorithms wherein a dynamic relationship is maintained between elements and processors, with elements having perhaps several processors operating on them simultaneously, and with elements disposing of their processors when they are no longer required, so that they may be re-used by other elements.

8 Future Plans

We will continue to apply this technique in the construction of parallel algorithms for a variety of quadtree tasks. In addition, we point out that we presented this technique as an embodiment of a control mechanism which can exploit fine-grained parallelism to create a useful dynamicism between processors and elements of our processing domain. In the future we plan to expand on the notion of this sort of dynamicism and apply it to other data structures and problem domains.

References

[1] John E. Warnock, "A Hidden Line Algorithm for Halftone Picture Representation", Technical Report 4-5, Computer Science Department, University of Utah, Salt Lake, May 1968

[2] Hanan Samet and Robert E. Webber, "Hierarchical Data Structures and Algorithms for Computer Graphics", *IEEE Computer Graphics and Applications*, May 1988

[3] Gregory Hunter and Kenneth Steiglitz, *Operations on Images Using Quad Trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, April 1979

[4] W. Daniel Hillis, *The Connection Machine*, M.I.T. Press, Cambridge, MA, 1985, Section 6.3

[5] J.D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, pp. 565-568, Addison-Wesley, Reading, MA, 1982

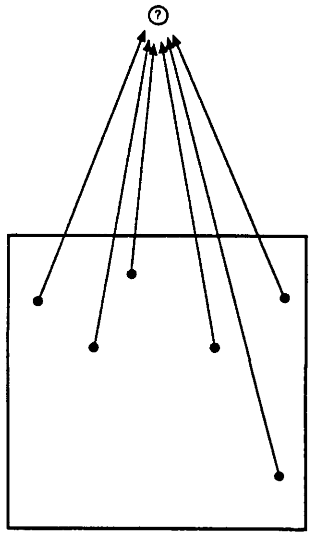


Figure 1: Before the first iteration

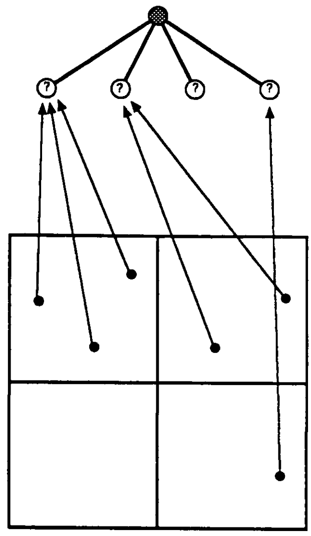


Figure 2: After the first iteration

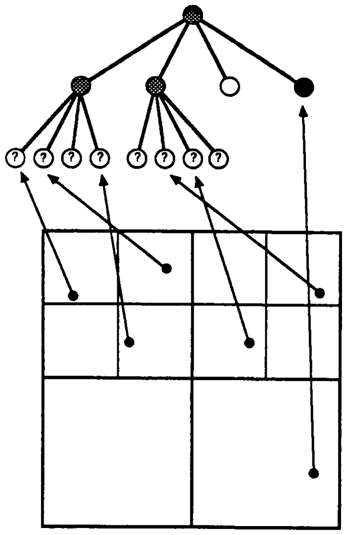


Figure 3: After the second iteration

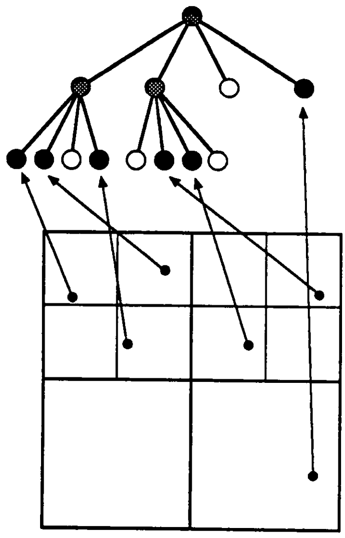


Figure 4: After the final iteration

ISBN 0-944426-38-7
ISBN 0-944426-44-1