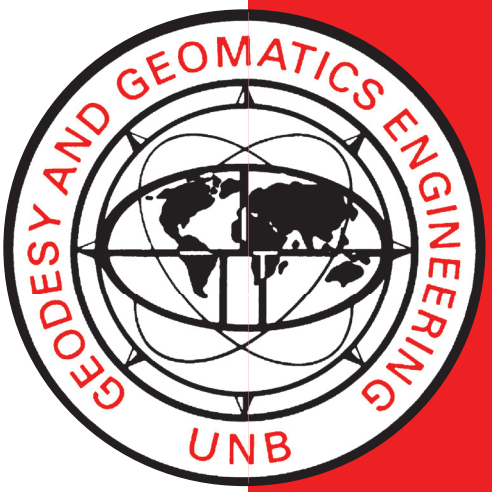
TRENDS AND CONCERNS OF SPATIAL SCIENCES

Y. C. LEE

June 1988



TECHNICAL REPORT
NO. 136

PREFACE

In order to make our extensive series of technical reports more readily available, we have scanned the old master copies and produced electronic versions in Portable Document Format. The quality of the images varies depending on the quality of the originals. The images have not been converted to searchable text.

TRENDS AND CONCERNS OF SPATIAL SCIENCES

PROCEEDINGS OF THE SECOND INTERNATIONAL SEMINAR

Chaired by

Y.C. Lee

Held at

FREDERICTON, NEW BRUNSWICK, CANADA
29 June to 1 July 1987

Sponsored by the

University of New Brunswick
University of Maine
Exchange Program

Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, New Brunswick
Canada
E3B 5A3

June 1988

PREFACE

This seminar series started in 1986 when Andrew Frank, of the University of Maine, invited a group of researchers to discuss issues related to spatial databases. The relatively small size and informal nature of the seminar encouraged interaction among the participants and generated many interesting conversations. It was decided to host the second seminar at the University of New Brunswick, and it was christened the Second International Seminar on Trends and Concerns of Spatial Sciences. It is encouraging to hear that the University of Laval will host the third seminar. From there, the University of Maine, the Univeristy of New Brunswick, and the University of Laval will take turns to organize this annual seminar.

The format of this second seminar is different from most conferences in an important way — each presentation lasted one-and-a-half hours to allow ample time at the end of the talk for discussions. These discussions were taped, transcribed, and included in the proceedings.

As summarized by Andrew Frank at the end of the seminar, the ten papers presented in this seminar covered three general areas: information storage, information representation, and information extraction. Max Egenhofer addressed the problem of storing an arbitrarily large spatial object as a coherent entity in the database. Marinos Kavouras, on the other hand, discussed the storage and representation of solids using octrees. Dave Armstrong represented the views of a data creator and a data user, and raised the question: What is the most useful structure for distributing topographic data? It seems that a hierarchical organization of structures with varying degrees of complexity is desirable.

Andrew Frank's paper pointed out the importance of formalizing 'naive' geometry, an intuitive approach to geometry we use daily to deal with spatial problems. The two papers on expert systems by Brad Nickerson and Vince Robinson, respectively, reviewed expert systems and their applications in GIS. A number of concerns were raised in this regard. One of them is the ease of handling spatial knowledge by current

expert systems, and the other is how spatial uncertainty can be modelled. Frank's 'naive' geometry touches on 'fuzzy' geometry, a concept which challenges the spatial expert system designers. User interface to a spatial database has always been a concern. Max Egenhofer and Andrew Frank presented an extension of SQL with spatial operators.

The problems of generating and manipulating curves and surfaces were addressed by Y.C. Lee and Chris Gold. Piecewise cubic polynomials have been used extensively in computer graphics but are relatively neglected in computer mapping systems. During the discussions, there were suggestions that mathematical curves might help generalization and the convex hull properties of B-splines and their like might serve useful purposes in geometric processing. Chris Gold pointed out that the traditional distinction of points, lines, and areas on a map is scale-dependent and is rather arbitrary. Hence there is a need for a generalized approach to the generation of Voronoi diagrams for spatial objects regardless of their representations. Convex hulls and their constructions were covered in detail by Joe Horton.

This second seminar was jointly funded by grants from the University of New Brunswick/University of Maine Exchange Program and from the Department of Energy, Mines and Resources Canada. For this assistance we are deeply grateful. We would also like to express our gratitude to the following who helped in various ways during the seminar: Dave Coleman, Roger Dick, Innocent Ezigbalike, Jimmy Loh, S.T. Tan, and Matthias Uhlenbruck. Finally, these proceedings would not have been finished without the expert help of Wendy Wells and Michael Fellows.

Y.C. Lee
Seminar Chairman

TABLE OF CONTENTS

OVERVIEW OF EXPERT SYSTEMS

Bradford G. Nickerson
School of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3
BGN@UNBMVS1.bitnet

An expert system can reasonably be defined as

> A computer program that uses heuristic knowledge and inference procedures to solve problems difficult enough to require significant human expertise.

Although identified as a computer program, expert systems have an organization distinctly different from other computer programs. It is this organization and its consequences which are discussed in this paper.

Background

The methods of expert systems have grown from research in the area of artificial intelligence. Game playing and puzzle solving on the computer led to the realization that many problems can be formulated in terms of **state space search**. The three ingredients of this state space search approach are:

1. A starting state (e.g., the initial state of a chess board).
2. A test for detecting final states of solutions to the problem (e.g., the rule for detecting checkmate in chess).
3. A set of operations that can be applied to change the current state of the problem (e.g., the legal moves of chess).

A problem is commonly represented as a search graph where each node of the graph corresponds to a particular state, and the directed arcs of the graph correspond to the legal operations on that state. To avoid performing an exhaustive search on the search graph, a so-called **heuristic** search is commonly employed to guide the choice of which operation to perform next when at a node.

The heuristic search uses an evaluation function $f(n)$ at node n to measure the progress towards the solution. As an example of an evaluation function, consider the problem of solving the '8-puzzle.' Figure 1 shows the initial and the desired position of the 8-puzzle. A simple evaluate function $f(n)$ for the 8-puzzle is $f(n) = g(n) + h(n)$, where $g(n)$ = cost of the path in the search graph from the start node to node n, and $h(n)$ = number of misplaced titles in the 8-puzzle associated with node n of the search graph. By choosing the next node with the lowest value evaluation function, the search for a solution is guided more directly than if an exhaustive search is done. The function $h(n)$ is called the **heuristic function**. A wise choice of this function leads to a much more direct path through the search graph to the solution.

Figure 1. (a) Initial position of the 8-puzzle.
(b) Desired position of the 8-puzzle.

The first system widely recognized as employing techniques typical of expert systems is called DENDRAL. The DENDRAL project began in 1965 at Stanford University. Its task is to assist an expert in determining the molecular structure of an unknown organic compound by interpreting the data obtained from a mass spectrometer. A mass spectrometer is a piece of laboratory equipment for chemical analysis which produces a set of peaks called a spectrum from a small sample of a given compound supplied in a gaseous form. The chemical sample is bombarded with a beam of electrons which

causes the compound to fragment and its components to become rearranged. Positively charged fragments are then collected by mass by passing the pieces through a magnetic field, and are displayed in a histogram which plots mass against relative abundance. The problem is that a complex molecule can fragment in more than one way.

DENDRAL uses three functional parts to handle the problem as follows [Barr and Feigenbaum, 1982]:

1. **PLAN**. The problem is constrained to be consistent with the results from the mass spectrometer. The constraints are listed in two parts; molecular fragments (clusters of atoms) that **must** be in the final molecular structure and fragments that **must not** appear in the final structure.

2. **GENERATE**. The DENDRAL algorithm generates only those molecular structures that do not include forbidden molecular fragments or exclude mandatory molecular fragments.

3. **TEST**. This part of DENDRAL ranks the resulting list of candidate structures by simulating their behavior in a mass spectrometer. Those structures resulting in simulated spectra closest to the empirical one are ranked highest.

DENDRAL has incorporated knowledge about mass spectrometry using rules. Besides the mass spectrum, DENDRAL is also given the atomic constituents of the molecule as input which allows it to infer the molecular weight, M, of the molecule. One of the rules used in the PLAN step which includes M is as follows:

If the spectrum for the molecule has two peaks at masses $x1$ and $x2$ such that

a. $x1 + x2 = M + 28$, and

b. $x1 - 28$ is a high peak, and

c. $x2 - 28$ is a high peak, and

d. at least one of $x1$ or $x2$ is high,

then the molecule contains a ketone group.

Any mass spectrum then meeting the requirements for this rule will have a ketone group as a mandatory constituent.

To date, DENDRAL has produced a number of results significant to chemists. These results have shown that it is possible for a computer program to equal the performance of experts in a very specialized area of science. Today DENDRAL runs on a computing facility at Stanford University and is accessed by hundreds of users daily. DENDRAL was one of the first programs to demonstrate the power of encoding domain-specific, heuristic knowledge separate from the control.

Expert systems today normally have the following three characteristics:

1. In general, they deal with a specific focussed task having a relatively narrow range of applicability.
2. Knowledge is kept separate from the reasoning method method to draw conclusions.
3. They are able to explain their own actions and lines of reasoning.

Central to the make-up of any expert system is the existence of an inference mechanism which arrives at conclusions by reasoning from rules. It is the separation of this so-called 'inference engine' from the knowledge base which makes expert systems more versatile than other computer programs. The knowledge base can be modified independent of the inference engine. The major parts of an expert system are shown in Figure 2.

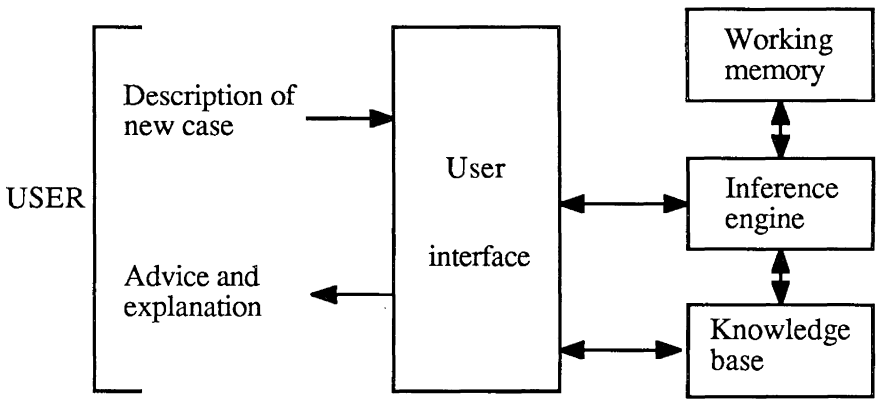


Figure 2. Parts of an expert system.

Knowledge Representation

The knowledge base is where the expert system stores both facts and rules. Codifying an expert's knowledge in form of **rules** was first implemented in DENDRAL, but another project at Stanford University was the first to be a (more or less) completely rule-based system which was unsupported by a clean, scientific representation. This system is called MYCIN (after the common suffix associated with many antimicrobial agents) and it successfully diagnoses and suggests therapy for bacterial infections. MYCIN embodies all of the features that have become the essence of today's expert systems. A further discussion of MYCIN is given below. Besides rules, some systems (e.g., CENTAUR, a consultation program for pulmonary diseases [Aikins, 1983]) also incorporate the notion of **frames** for knowledge representation [Minsky, 1975]. Both of these methods are now considered.

An example of a simple rule grammar expressed in Backus-Naur Form (BNF) is as follows:

```
<rule> ::= (IF {<antecedent>} + THEN {<consequent>}+)
<antecedent> ::= <associative-triple> | <associative-triple>  AND
<consequent> :: <associative-triple>
<associative-triple> :: (<attribute> <object> <value>)
```

where <attribute>, <object>, and <value> are domain specific terms. As an example of a rule expressed using this grammar, consider the following:

```
(IF (area lot39 0.3)  AND  (location lot39 city)
THEN (taxrate lot39 0.9))
```

This rule means that if lot39 is of area 0.3 hectares and is located in the city, then the tax rate for this lot is 0.9% of the assessed value. Suppose we wish to make the rule more general and have it say that any lot in the city with an area between 0.2 and 0.8 hectares is to have a tax rate of 0.9%. Such a rule is

```
(IF (area *lot *number)  AND  (location *lot city)  AND
      (greater-than *number 0.2)  AND  (less-than  *number 0.8)
THEN (taxrate *lot 0.9))
```

Here we have introduced two variables, *lot and *number, which do not denote particular objects, but are used as place holders that will match with suitable values.

The three occurrences of the *number variable must be **instantiated** to the same value when this rule is invoked. Similarly, the *lot variable must be instantiated to the same value in both the antecedent and the consequent, regardless of which one is instantiated first.

The example grammar given above is normally not adequate to represent rules for a real application. MYCIN also provides a so-called **certainty factor** with each argument. Both MYCIN and DELTA (an expert system for diesel electric locomotive repair [Bonissone and Johnson, 1984]) allow the use of predicate functions to better describe the rule antecedents. Rule grammars are normally given in a BNF notation to formalize the rule definition process, and they commonly include the <attribute> <object> <value> construct.

Frames are a way of grouping information in terms of a record of **slots** and **fillers**. Frames are essentially a data structure for representing typical situations. An example is taken from the Hayes-Roth et al. [1983] description of RLL (representation-language language). Figure 3 shows an example of some frames representing a drainage basin which includes several manholes.

The first frame in Figure 3 (M6-3) represents the third manhole in drainage basin six. The third frame represents the set of all manholes. The final pair of frames represent quite abstract knowledge. The FeedsInto frame defines itself as a slot. Any manhole m1 may have a FeedsInto slot. If it does, it must be filled by the name of another manhole m2 (the Datatype slot). The Inverse slot implies that the m2 frame should list m1 as one of the entries in its FeedsFrom slot. To compute which manhole m1 feeds into, first locate all the downstream pipes, find their ends, and return one of those values (other than m1 itself).

In general, frames provide a method of representing prototypical situations. The slots and their expected values tie together knowledge about a given situation and provide expectations about what objects will be present and what events will occur in the situation. An expert system called CENTAUR [Aikins, 1983] uses both frames and production rules to interpret measurements from lung function tests of patients. The frames in CENTAUR represent the expected lung test results for one pulmonary disease

or its subtype. Aikins concluded from her research that frames are more suitable structures than rules for representing **standard data patterns** with ranges of plausible data values.

M6-3	
Isa:	Manhole
FeedsInto:	M6-2
LocatedUnder:	(MainStreet and OakDrive

M6-2	
Isa:	Manhole
FeedsFrom:	(M6-3)
LocatedUnder:	(MainStreet and RidgeRoad)

Manhole	
Isa:	Set
Generalizations:	(PhysObject, CircularObject, Artifact)
Examples:	(M6-2, M6-3, ...)
Description:	This represents the collection of all manholes
Prototype:	TypicalManhole
NewSlots:	(FeedsInto, FeedsFrom, LocatedUnder)

LocatedUnder:	
Isa:	Slot
Description:	The street(s) above the manhole
Format:	List
Datatype:	Streetname
MakesSenseFor:	(Manhole)
Inverse:	LocatedAbove

FeedsInto	
Isa:	Slot
Description:	Whatever is directly downstream
Inverse:	FeedsFrom
MakesSenseFor:	(Manhole)
HighLevelDefn:	(Compose: OtherEnd DownstreamPiles)
Format:	Singleton
Datatype:	Manhole
ToCompute:	(lambda (m) (OneOf (Remove m (Ends (DownstreamPipes m)))))

Figure 3. Example of frames [Hayes-Roth et al., 1983].

Inference Engine

The so-called inference engine of a rule-based expert system can be described in terms of the 'recognize-act' cycle. It consists of the following sequence of steps:

1. Match the rule patterns against elements in working memory.

2. If there is more than one rule that could fire, then decide which one to apply using the 'conflict resolution' strategy.

3. Apply the rule (which might add or delete an element to or from working memory) and then go to step 1.

Working memory contains all of the presently known facts (often in the form of <object> <attribute> <value> triples) about the problem being solved. It usually is initialized to contain a single hypothesis or fact to start the inference engine mechanisms.

In step (2) above, there may be more than one rule which matches with the facts in working memory. This matching process binds constants to variables in the rules resulting in instantiations. The entire set of instantiations involved with the rules which presently match working memory facts is called the **conflict set**. An example of an instantiation is the pair {*lot/lot39} for the example rule given above. The following are three conflict resolution strategies used in practice [Jackson, 1986]:

1. Refractoriness: A rule is not allowed to fire more than once on the same data. To implement this, all instantiations which have been encountered before are deleted from the conflict set.

2. Recency: A time tag on the elements in working memory tells when they were added. Rules which use more recent data are preferred to rules which match against data that has been in working memory for a longer time.

3. Specificity: Rules which have a greater number of antecedent conditions are preferred to more general rules with fewer antecedent conditions. As an example, given the two rules

   [rule1] (IF (location *lot YorkCounty)
   THEN (location *lot rural))

   [rule2] (IF (location *lot YorkCounty) AND (location *lot Fredericton)
   THEN (location *lot city))

   one would choose [rule2] since it is more specific.

The inference engine normally uses one of three control mechanisms: (1) forward chaining, (2) backward chaining, or (3) a combination of both forward and backward chaining. Forward and backward chaining are also called data driven and goal driven control, respectively.

Forward chaining control seeks to satisfy antecedents before asserting the rule consequents. To start the process, the known facts about the problem must be supplied

to working memory. The FCHAIN pseudo-Pascal procedure shown below is a simplified version of how the forward-chaining control scheme works. The CONFLICT_RESOLVE procedure chooses one rule *R* from the set *S* using one or more of the conflict resolution techniques mentioned above.

```
procedure FCHAIN;
scan working memory to find the set S of rules whose antecedents are satisfied;
while S is nonempty and the problem is unsolved do
    begin
        CONFLICT_RESOLVE(S,R);
        apply R to update working memory with R's consequents;
        scan working memory to find the set S of rules whose antecedents are satisfied
    end
end;
```

Although not shown, there must also be a mechanism to handle the case where there are no applicable rules and the problem is still unsolved. In this case, the system will normally choose some rule which is nearly satisfied and ask the user to confirm or deny any unsatisfied antecedents.

Backward-chaining control first chooses a goal or hypothesis to be proven. This goal is then searched for in the consequents of rules. If at least one rule is found with the desired goal as a consequent, then the rule antecedents are recursively evaluated. The following simplified procedure describes the backward-chaining process to determine a goal *G*:

```
procedure BCHAIN(G);
if G is in working memory then G is true
else
   begin
      scan working memory to find the set S of rules whose consequents
      include G;
      if S is empty then ask the user about G else
         begin
            CONFLICT_RESOLVE(S,R);
            for each antecedent G' of R do BCHAIN(G');
            if all antecedents of R are true then
               begin
                  add all consequents of R to working memory;
                  G is true
               end;
            else G is false
         end
   end
end;
```

The conflict resolution scheme used by BCHAIN must decide which rule to choose given that more than one rule contains the present goal *G* as a consequent. Backward chaining has the advantage that it asks questions related to its overall goal. In addition, a reasonable explanation of the behavior of the system can be given simply by telling the user what goal it is working on and what rules it is using. An example of forward and backward chaining is shown in Figure 4.

```
WM : (area lot39 0.4) (position lot39 N432116W662125)
[rule1]   (IF (position *lot *pos) AND (inside location *pos city)
          THEN (location *lot city))
[rule2]   (IF (area *lot *number) AND (location *lot city) AND
                (> *number 0.2) AND (< *number 0.8)
          THEN (taxrate *lot 0.9))
```

(a)

```
1. S = {[rule]}; R = [rule1]; {*lot/lot 39, *pos/N432116W662125};
   (inside location N432116W662125 city) supplied by a PIP test.
2. WM = ((area lot39 0.4) (position lot39 N432116W662125) (location lot39 city))
3. S = {[rule1],[rule2]}, but [rule1] already used => R = [rule2] with {*number/0.4}.
4. WM = ((area lot39 0.4) (position lot39 N432116W662125)
   (location lot39 city) (taxrate lot39 0.9))
5. Problem is solved with taxrate = 0.9.
```

(b)

```
1. G = (lot39 taxrate *trate)
2. S = {[rule2]}; R = [rule2]; BCHAIN({antecedents of R}).
3. Antecedents proved with {*lot/lot39,*number/0.4} except (location lot39 city).
4. G = (location lot39 city)
5. S = {[rule1]}; R = [rule1]; BCHAIN({antecedents of R}).
6. Antecedents proved with {*pos/N432116W662125}.
7. WM = ((area lot39 0.4) (position lot39 N432116W662125) (location lot39 city))
8. Original goal is true with {*trate/0.9}.
```

(c)

Figure 4. An example of forward and backward chaining.
(a) Initial working memory (WM) and rules. (b) Forward chaining solution.
(c) Backward chaining solution.

Notice that [rule1] above includes a predicate called 'inside' which invokes a point in polygon (PIP) test to determine if a given position falls inside the city boundary.

Expert systems are often used to represent uncertain information. A common approach is to use a so-called **certainty measure** $C(A)$ associated with each assertion A. If $C(A) = 1$, then A is known to be true; for $C(A) = -1$, then A is known to be false; for $C(A) = 0$, then we know nothing about A. Similarly there is a **certainty factor** CF associated with every rule. Initially, the certainty associated with any assertion is 0.

If a rule says that $B \rightarrow A$ with certainty factor CF, then the certainty of A is changed to CF when B is observed to be true.

What happens when $C(A)$ is non-zero, and we wish to apply $B \rightarrow A$ with CF knowing that B is true? We then compute $C(A \mid B)$ (certainty of A given B) as follows [Buchanan and Duda, 1983]:

$$C(A \mid B) = \begin{cases} CA + CF - CA*CF & \text{if } CA \text{ and } CF > 0, \\ CA + CF + CA*CF & \text{if } CA \text{ and } CF < 0, \\ \dfrac{CA + CF}{1 - \min[\,|CA|\,,\,|CF|\,]} & \text{otherwise.} \end{cases}$$

Using this formula repeatedly, we can combine the conclusions of any number of rules $B1 \rightarrow A, B2 \rightarrow A, \ldots, Bn \rightarrow A$. The formula has the following properties:

1. The resulting $C(A \mid B)$ always lies in the range (-1, 1), being +1 if CA or CF is +1, and -1 if CA or CF = -1.
2. When contradictory conclusions are combined (i.e., $CA = -CF$), then $C(A \mid B) = 0$.
3. Except for the singular points (-1,1) and (1,-1), $C(A \mid B)$ is a continuous function of CA and CF, increasing monotonically in each variable.
4. The formula is symmetric in CA and CF.
5. The results of the formula when more than two pieces of evidence are combined are independent of the order in which the evidence is considered.

When the premise B for a rule $B \rightarrow A$ is itself uncertain, then the strength of the conclusion should be reduced. This is done by multiplying CF for the rule by the certainty of B, provided that $C(B)$ is positive. If $C(B)$ is negative, the rule is considered inapplicable and is not used.

In general, the premise B of a rule is a conjunction of antecedents. Since each antecedent Bi can have its own certainty value, then the certainty of the conjunction is computed as

$$C(B1 \text{ AND } B2) = \min[C(B1), C(B2)] \ .$$

If the premise is a disjunction of antecedents, then the certainty measure is computed as

$$C(B1 \text{ OR } B2) = \max[C(B1), C(B2)] \ .$$

Certainty measures are not the only way to deal with uncertainty in expert systems. Bayesian probability theory, Zadeh's theory of possibility, and the Dempster-Shafer theory of evidence have all been tried. In general, these other methods are more complicated to implement than the certainty measure approach, and are not as widely used.

User Interface

A very important part of an expert system is the user interface. Normally, a large effort is made to provide a friendly, often graphics-oriented interface which the users will find comfortable. Besides this, expert systems often provide an explanation facility which can answer questions about the problem being solved. In particular, users can inquire **why** a question is being asked of them, or **how** a particular conclusion was established by the expert system. This provides a positive feedback for the expert user so that he or she can fully understand the reasoning process. It also serves as a useful debugging tool in the development phase of an expert system.

A Case Study: MYCIN

Most of this section is taken from Buchanan and Shortliffe [1984]. The MYCIN project began in 1972 at Stanford University with a set of discussions between medical school and computer science researchers interested in applying more intelligence to computer programs that interpret medical data. MYCIN was designed to provide advice to physicians through a consultative dialogue. The end goal of a consultation is to provide an appropriate choice of antimicrobial (antibiotic) therapy for a patient suffering from a bacterial infection. In 1979, MYCIN was evaluated on its selection of antimicrobial agents for actual cases of acute infectious meningitis. The evaluation was performed in a 'double blind' fashion using two phases. In the first phase, ten patient cases were evaluated by ten **prescribers** (one of which was MYCIN). In the second phase, eight **evaluators** assessed these prescriptions without knowing the identity of the

prescribers. Overall, MYCIN was rated by the evaluators as better than all other nine prescribers. All eight evaluators and most of the prescribers (there was one resident and one student prescriber) were infectious disease specialists. From this evaluation one can conclude that MYCIN is truly an expert in its specialized field.

MYCIN is a backward-chaining rule-based expert system. It is written in LISP and presently contains approximately 500 rules. To get a feeling for how MYCIN works, Figure 5 shows a portion of an actual consultation.

A portion of its MYCIN's rule grammar is as follows:

```
<rule>      ::=  <premise> <action>
<premise>   ::=  ($AND {<condition>}+)
<condition> ::=  (<func1> <context> <parameter>) |
                 (<func2> <context> <parameter> <value>) |
                 (<special-func> <arguments>) |
                 ($OR {<condition>}+)
<action>    ::=  <concpart>
<concpart>  ::=  <conclusion> | <actfunc> |
                 (DO-ALL {<conclusion>}+) |
                 (DO-ALL {<actfunc>}+1)
```

Not all of the non-terminals are defined, but enough of the grammar is given to see that the premise of a rule is a conjunction of conditions, where a condition itself may be a disjunction of conditions. The <func1>, <func2>, and <special-func> non-terminals are predicates. As an example, the predicate SAME is of type <func2>, and an example specification for it is

```
(SAME CNTXT SITE BLOOD)
```

It follows the <object> <attribute> <value> construct with the addition of a beginning <predicate> non-terminal. An example of a complete rule in its internal LISP structure is

1) Patient's name: (first-last)
** **FRED BRAUN**
2) Sex: *(Questions 1-3 request*
** **M** *background patient data.)*
3) Age:
** **55**
4) Are there any cultures for Fred Braun which may be related to the present illness, and from which organisms have been grown successfully in the microbiology laboratory?
** **Y**

--------CULTURE-1--------
5) From what site was the specimen for CULTURE-1 taken?
** **BLOD**
=**BLOOD** *(System corrects typing error.)*
6) Please give the date and time when CULTURE-1 was obtained. (mo/da/yr time)
** **JUN 20, 1977**
The first organism isolated from the blood culture of 20-JUN-77 (CULTURE-1) will be referred to as:

--------ORGANISM-1--------
7) Enter the laboratory-reported identity of ORGANISM-1:
**UNKNOWN *(Typically identity is not yet known.)*
8) The stain (gram or Ziehl-Neelsen acid-fast) of ORGANISM-1:
** **NEG** *(Preliminary lab results give some clues.)*
9) Is ORGANISM-1 rod or coccus (etc.):
** **ROD**
10) What is the form of the individual organisms (e.g. lancet-shaped for cocci, fusiform for rods, etc.)?
** **FUSIFORM**

(. . . more questions follow in order to gather sufficient information to infer the identity and significance of the infecting organisms . . .)

23) What is the suspected portal of entry of the gramneg rod from the blood culture of 20-JUN-77 (ORGANISM-1) into this sterile site?
** **GI (6)** *(User indicates the uncertainty of his response.)*
24) Is Fred Braun a burn patient?
** **N**
25) Is Fred Braun a compromised host (e.g. alcoholic, sickle-cell-disease, immunosuppressed)?
** **N**
26) Is Fred Braun febrile?
** **Y**

(. . . after about 50–60 questions, MYCIN prints the diagnostic hypotheses on which therapy will be based . . .)

My therapy recommendations will be designed to treat for organisms that are either very likely or, although less likely, would have a significant effect on therapy selection if they were present. It is important to cover for the following probable infection(s) and associated organism(s):

INFECTION-1 is BACTEREMIA
<ITEM-1> E.COLI [ORGANISM-1]
<ITEM-2> KLEBSIELLA [ORGANISM-1]
<ITEM-3> ENTEROBACTER [ORGANISM-1]
<ITEM-4> KLEBSIELLA-PNEUMONIAE [ORGANISM-1]

(. . . questions follow to evaluate possible therapy choices, and finally MYCIN prints its therapy recommendations . . .)

[REC-1] My preferred therapy recommendation is as follows:
In order to cover for items <1 2 3 4>:
Give: GENTAMICIN *(One drug covers 4 possible identities.)*
Dose: 119 mg (6.0 ml) q8h IV for 10 days [calculated on basis of 1.7 mg/kg]
Comments: Modify dose in renal failure.

Figure 5. Excerpts from a MYCIN consultation [Buchanan and Shortliffe, 1984]. Note that comments in italics are not part of the actual interaction.

```
PREMISE:  ($AND    (SAME CNTXT GRAM GRAMNEG)
                   (SAME CNTXT MORPH ROD)
                   (SAME CNTXT AIR AEROBIC))
ACTION:   (CONCLUDE CNTXT CLASS ENTEROBACTERIACEAE TALLY .8)
```

When translated into English, this rule reads as follows:

IF: 1) The stain of the organism is gramneg, and
2) The morphology of the organism is rod, and
3) The aerobicity of the organism is aerobic,

THEN: There is strongly suggestive evidence (.8) that the class of the organism is enterobacteriaceae.

The translation of the LISP rules into English is facilitated by storing a translation property for every clinical parameter (e.g., GRAM), function (e.g., SAME), and other pieces of the MYCIN knowledge base. These translation properties contain the English strings and the way in which they should be combined to make a coherent statement. MYCIN's rules are organized into 12 categories. Every rule belongs to one and only one of these categories.

Besides rules, MYCIN has a separate class of second-level knowledge called **clinical parameters** which is vital to its operation. There are currently 65 clinical parameters, each with an associated set of properties. Examples of clinical parameters include the name of a patient, the site of a culture, the morphology of an organism, and the dose of a drug. Besides the translation property mentioned above, each parameter has a list of all the rules in which the <action> clause permits a conclusion to be made regarding the value of the clinical parameter.

MYCIN is controlled by a four-stage process as follows:

1. Decide which organisms, if any, are causing significant disease.
2. Determine the likely identity of the significant organisms.
3. Decide which drugs are potentially useful.
4. Select the best drug or drugs.

This process is started using a single rule called the **goal rule** whose English translation is as follows:

RULE092

IF: 1) There is an organism which requires therapy, and
2) consideration has been given to the possible existence of additional organisms requiring therapy, even though they have not actually been recovered from any current cultures

THEN: Do the following:
1) Compile the list of possible therapies which, based upon sensitivity data, may be effective against the organisms requiring treatment, and
2) determine the best therapy recommendations from the completed list

OTHERWISE: Indicate that the patient does not require therapy

When MYCIN first tries to evaluate the premise of this goal rule, the first condition requires that it know whether there is an organism that requires therapy. MYCIN then reasons backwards using its rules. This reasoning process is controlled by two procedures called MONITOR and FINDOUT (see Figures 6 and 7).

Notice that MONITOR calls FINDOUT if it cannot determine the value of a clinical parameter which is necessary to evaluate the premise of a rule. FINDOUT then attempts to obtain the missing information, which may result in the firing of another rule whose premise is evaluated by MONITOR. This sets up the recursion resulting in the backward chaining of MYCIN's rules.

The property of clinical parameters known as LABDATA is a flag used in the first decision box of FINDOUT. If true, this means that this parameter is a piece of 'primitive' data that is always requested from the user (e.g., the patient's age). If LABDATA is false, FINDOUT attempts to determine the value of the clinical parameter by using a list of rules. This rule list *Y* is exactly the list mentioned above which contains all rules which can make a conclusion about this clinical parameter. Instead of searching through working memory to find the rules whose consequents can determine this clinical parameter, MYCIN already knows which rules are appropriate.

THE MONITOR FOR RULES

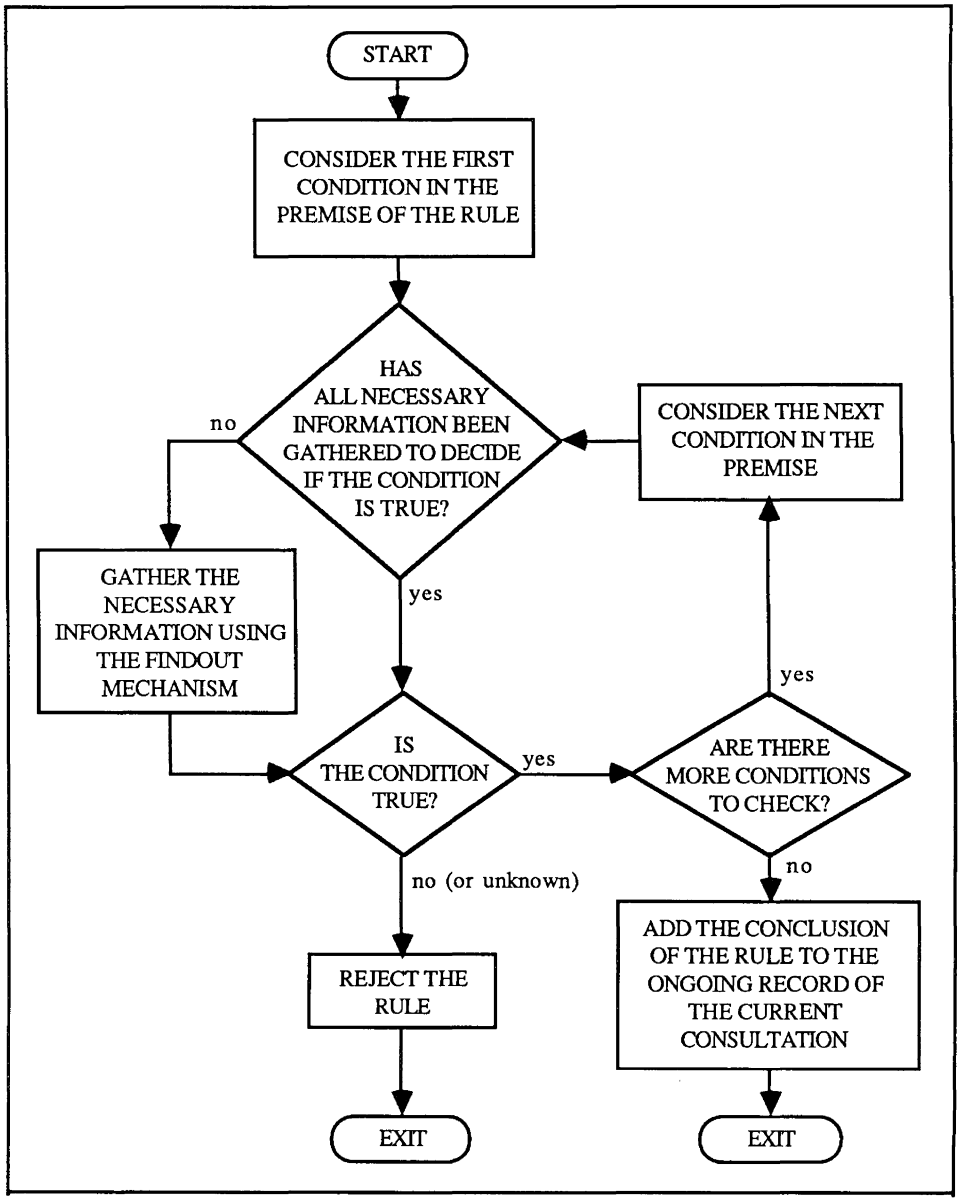


Figure 6. Flowchart showing how the MONITOR analyses a rule and decides whether or not it applies to the present situation [Buchanan and Shortliffe, 1984].

THE FINDOUT MECHANISM

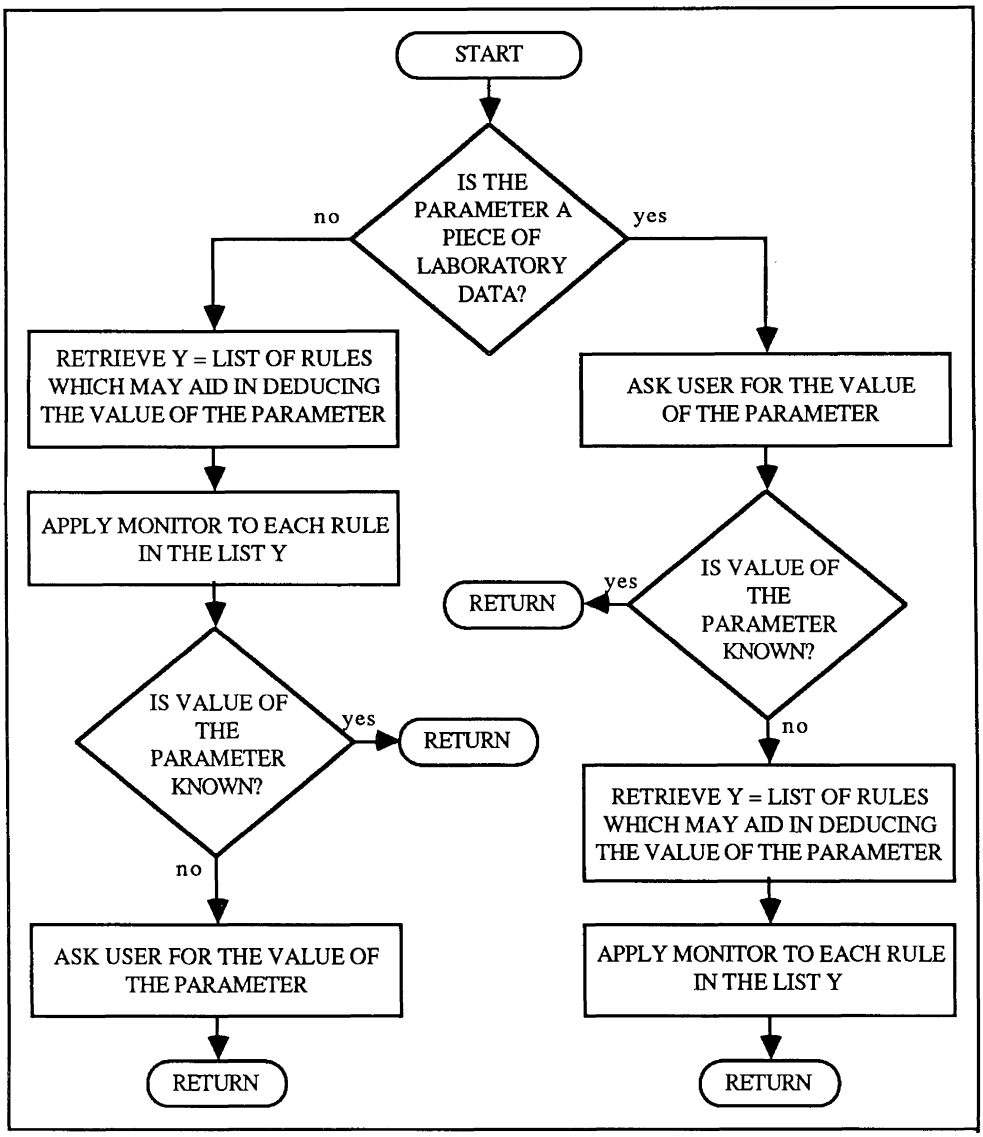


Figure 7. Flowchart of FINDOUT for determining which questions to ask the physician. The derivation of values of parameters may require recursive calls to the MONITOR, thus dynamically creating a reasoning chain specific to the patient under consideration [Buchanan and Shortliffe, 1984].

From the above discussion, it is evident that the order of conditions within a premise is highly important. In general, conditions referencing the most common parameters (i.e., those that appear in the premises of the most rules) are put first in the premises of new rules to act as a screening mechanism.

After MYCIN determines that the premise of RULE092 is true, it must then select an appropriate therapy based on its diagnosis. Besides the rules for diagnosis determination, MYCIN has another set of rules which appears as follows:

IF: The identity of the organism is pseudomonas
THEN: I recommend therapy chosen from among the following drugs:
1 - colistin (0.98)
2 - polymyxin (0.96)
3 - gentamicin (0.96)
4 - carbenicillin (0.65)
5 - sulfisoxazole (0.64)

The numbers associated with each drug are the probabilities that a specific organism will be sensitive to the indicated drug based on tests performed at Stanford Hospital. There is one such rule for every kind of organism known to MYCIN. Besides these rules, MYCIN performs complicated reasoning to ensure that the therapy it recommends is the **best** one based on other drug-specific knowledge.

Out of necessity, there is a good deal of information on how MYCIN works omitted from the above description. In summary, it is seen that MYCIN contains a good deal of knowledge besides its rules, although the bulk of the logic used by a physician to determine a diagnosis and therapy are in these rules. The initial coding of this logic based on verbal explanations from physicians was not a trivial task for MYCIN since the rules are stored as LISP code using a fairly complicated grammar. MYCIN does demonstrate that a primarily rule-based paradigm is a very powerful way of encoding and maintaining an expert's knowledge.

Conclusions

MYCIN is not the only successful, widely recognized expert system from which much can be learned. R1 (also called XCON) is a rule-based expert system used by

Digital Equipment Corporation to configure its computers for customers [McDermott, 1982; Bachant and McDermott, 1984]. It is a forward-chaining expert system written in the OPS5 programming language, which, in turn, is written in LISP. The expert it emulates is called a technical editor. Technical editors know which detailed components to assemble to make a complete, e.g., VAX computer. As of 1984, R1 contained approximately 3300 rules and has knowledge of over 5500 parts making up DEC computers. In the fourth quarter of 1983, R1 configured more than 20 000 orders with 10% of these exhibiting problems of some kind.

Both MYCIN and R1 are successful expert systems performing at or above the levels of their human counterparts. A good deal of domain expertise is required to make the systems work well. As an example, Edward Shortliffe (who wrote the initial version of MYCIN for his Ph.D. dissertation) is also a medical doctor. The beauty is that once encoded into an expert system, this domain knowledge is forever captured and accessible to anyone knowledgeable in the field, expert or not.

The task of developing expert systems is being made easier with the appearance of many expert system shells which contain all of the control mechanisms, but lack any domain knowledge. An example of this is EMYCIN (Essential MYCIN), which is MYCIN without its domain knowledge. Many other commercial expert system tools are also now available [Gevarter, 1987].

Waterman [1986] reports over 180 expert systems in existence. Of these, only nine are classified as commercial systems, with the rest being prototypes of one sort or another. A good deal or work is required to develop and maintain a successful expert system. R1 requires a continuing development effort of approximately four man-years per year [Bachant and McDermott, 1984]. Waterman [1986] points out that it currently takes anywhere from five to ten man-years to build an expert system to solve a moderately complex problem.

The separation of knowledge from control is the main advantage of expert systems. Before jumping to the conclusion that an expert system is just what is needed to solve a particular problem, one must carefully consider whether an expert system approach is appropriate. For example, it appears that expert systems are not very good at

representing temporal or spatial knowledge [Waterman, 1986]. It can be concluded, though, that expert systems provide a viable alternative for tackling difficult problems by computer.

References

Aikins, J.S. (1983). "Prototypical knowledge for expert systems." *Artificial Intelligence*, 20, pp. 163-210.

Bachant, J. and J. McDermott (1984). "R1 revisited: four years in the trenches." *The AI Magazine*, fall, pp. 21-32.

Barr, A. and E.A. Feigenbaum, eds. (1982). *The Handbook of Artificial Intelligence*, Vol. 2, William Kaufman, Los Altos.

Bonissone, P.P. and H.E. Johnson (1984). "Expert system for diesel electric locomotive repair." In *Human Systems Management 4*, North-Holland, pp. 255-262.

Buchanan, B.G. and R.O. Duda (1983). "Principles of rule-based expert systems." In *Advances in Computers*, Vol. 22, M.C. Yovitts, ed., Academic Press, New York, pp. 163-216.

Buchanan, B.G. and E.H. Shortliffe, eds. (1984). *Rule-Based Expert Systems*, Addison-Wesley, Reading MA.

Gevarter, W.B. (1987). "The nature and evaluation of commercial expert system building tools." *IEEE Computer*, May, pp. 24-41.

Hayes-Roth, F., D.A. Waterman, and D.B. Lenat (1983). *Building Expert Systems*, Addison-Wesley, Reading, MA.

Jackson, P. (1986). *Introduction to Expert Systems*, Addison-Wesley, Wokingham, England.

McDermott, J. (1982). "R1: A rule-based configurer of computer systems." *Artificial Intelligence*, 19, pp. 39-88.

Minsky, M. (1975). "A framework for representing knowledge." In *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, pp. 211-277.

Waterman, D.A. (1986). *A Guide to Expert Systems*, Addison-Wesley, Reading MA.

Discussion

Question: What is the difference between a certainty factor and a probability factor?

Answer: A probability factor ranges from zero to one and it is based on statistical theory. A certainty measure was developed for MYCIN and does not necessarily comply with the statistical theory. It has a value of negative one to positive one. The negative value indicates a measure of disbelief in a hypothesis.

Comment: The paper by Babrow and Collins (editors 1975, *Representation and Understanding*, New York, Academia Press) discusses the types of problems that can be solved by an expert system. It strongly discourages the use of temporal data.

Comment: The main difficulty is that we can only mimic certain sound logical processes.

Comment: Does an expert always gives the right answer? No, even the builders of R1 made mistakes. In the MYCIN case study, among the experts that evaluated the ten patients, there is only about an 80% level of agreement on the treatment of the disease after it was diagnosed correctly.

Question: What is the legal implication of using an expert system, i.e.,can we sue an expert system?

Answer: That is one of the reasons why MYCIN and many other expert systems are being classified as research prototypes and not as commercial products.

Comment: A similar situation arose as the direct consequence of programming blunders in a program that operates a nuclear plant.

Comment: The main problem in spatial data analysis is that there are no clear rules to deduce information and very frequently, a lot of the terminology used is very subjective.

Comment: The point in polygon test is an example of spatial knowledge that can be clearly defined. This situation is only true for well defined polygons and may not be so for polygons with fuzzy edges.

Question: What kind of qualifications are required of a knowledge engineer?

Answer: Well, 'knowledge engineer' are words that are bandied about, but I'm not quite sure what they mean. Let me go back to MYCIN, and again mention that it was written by a medical doctor. A 'knowledge engineer' would be the person who would code the rules in LISP, who would know how the translation facility worked, and who would do the coding and the programming necessary to make it work. The 'knowledge engineer' would sit down with the expert and discuss many examples of what the expert does. In this way, the terminology, rules, and exceptions to the rules would eventually be encoded into the knowledge base. It's a very difficult job.

Comment: There is a problem that I would like to mention before we go on to the next topic, and in a way it was mentioned by your discussion of MYCIN. This is the organization of rules or the organization of knowledge. In general, we can build systems that have wider implications than expert systems, the same as for ordinary programming. We can build systems of arbitrary complexity, but we are limited in what we can do practically by the complexity. Two years ago at the conference we talked about building knowledge bases and expert systems (it is an interesting, sort of fuzzy idea about what is the difference between them) and people there agreed whole-heartedly that the limits are about 1000 rules.

Question: How do we know?

Answer: They were experts. People like Rolfs and Insberg, and similar people were there. They looked at their past experience and that's about as far as we can go.

Forty thousand rules is not feasible; we cannot handle it. XCON is the exception. XCON has very simple rules. A large set of the rules in XCON is very simple. They describe the components and that's a relatively simple task.

Comment: But XCON also has an organization. I mean, there are rules for the VAX and different rules for the PDP11/23 and PDP11/44. All those 3300 rules are subdivided into classes.

Comment: The problem is really how to organize the rules and that is what the knowledge engineers are good at. What they learn is how to structure knowledge such that every case is not a special case, but we have some large groups which can be entirely understood.

Comment: The use of frames is one attempt to do that. It represents a prototypical situation, but then you end up writing a whole bunch of frames to handle the exceptions, so there are problems with frames too. Some expert system tools combine frames and rules, like KEE for example. I have not actually used KEE, but I would suspect that would be a good way to do it. You could get knowledge that is prototypical into frames and exceptions into rules, some how.

Comment: So you would say that the broader class of tools you have, the more useful. I would try to contour it. I would say the restriction on a few well-understood methods is a better way to go, but I can't prove my point either.

Comment: The smaller the problem that the system works on, the better it will be able to handle it. If we want to create an Expert System for all spatial reasons, we will probably not get by.

Comment: But the point before, Max, was different. The point was, do we need tools which include forward and backward chaining, and a variety of other methodologies?

Comment: But remember, you do not need to use all those methods; you may only use one piece of it. That's the beauty of having a tool that can do everything—you only choose the parts that are applicable. If you only want to use backward reasoning, fine. I think that something like EMYCIN might be a good way to go in a first attempt at building a knowledge based expert system, for example, for spatial knowledge.

Question: Can you explain to me how EMYCIN seems to be able to deal with negative knowledge. It is well known that first-order knowledge is unsolvable, in the general case. How do they get around that? Where is the hole?

Answer: I have not actually seen a rule in EMYCIN that uses a negative inference. In other words, if such and such conditions are true, then we know the identity of the organism will be known with certainty -0.8. I haven't seen any examples of that. They do have the facility to do it though.

Comment: They would probably structure a rule then.

Answer: Yes, if they know that certain conditions indicate disbelief in an organism, the fact is written with a negative certainty factor.

Question: Yes, that is what I see. But I do not see where the mathematics are relatively rigorous, so that there must be somewhere else that is open.

Answer: Yes, the certainty measure, the formula that I showed you for propagating certain factors, has caused some discussion about its theoretical soundness. However, it seems to do the job and it is used in a lot of expert systems.

Question: Where is that discussion?

Answer: The reference is Buchanan, B.G. and E.H. Shortliffe (1984). *Rule-Based Expert Systems*. Addison-Wesley, Reading, MA, pp. 209-219 and pp. 263-271.

Comment: There are many other systems that can be used, such as possibility theory, probability theory, Bayesian inference, and the Dempster-Shafer theory. All of these tend to be more complex than the certainty factors.

End

AN OVERVIEW OF EXPERT SYSTEMS FOR GEOGRAPHIC INFORMATION SYSTEMS IN RESOURCE MANAGEMENT

Vincent B. Robinson[1]
Ness S.T. Lee
Department of Surveying Engineering
The University of Calgary
2500 University Drive NW
Calgary, Alberta
Canada
T2N 1N4
CA@UNCAEDU.BITnet

ABSTRACT

Resource management involves a complex planning and decision making process based on spatial information. Geographic information systems (GIS) are becoming recognized as powerful tools for resource management. GIS are computer-based systems for the collection, storage, management, query, and mapping of geographic data. Expert systems have been applied to some of the problems in GIS: feature extraction, database systems, decision support, and map design. There remains a large array of areas where use of expert systems would make GIS more directly responsive to the needs of resource managers. These include development of more intelligent user interfaces, better natural language interpretation of input/query, more efficient database search techniques, improved image classification, extraction of high-level geographic features from digital data, improved management of uncertainty, production of higher quality maps, and use of multiple knowledge bases.

Introduction

Current practices of resource management involve complex planning and decision making in a very short period of time. Much of the information required for resource

[1]As of 1 August 1988, Dr. Robinson will be Director of Institute of Land Information Management at University of Toronto, Erindale Campus, Mississauga, Ontario,L5L 1C6

management is spatial. Conventional information systems are not adept at handling spatially distributed information or at producing useful maps. Geographic Information Systems (GIS) are computer-based systems for collecting, storing, managing, retrieving, transforming, and displaying spatial data from the real world. For this reason, GIS are becoming recognized as powerful tools for resource management [Webb 1982]. A GIS is a computer-based system for collecting, storing, managing, retrieving, transforming, and displaying spatial data from the real world for a particular set of purposes.

The U.S. Bureau of Land Management has responsibilities for over 340 million acres of land and resources. As a result of the increasing complexity of their resource management problems, they are committed to using GIS technology to help manage mineral, range, wildlife, and forest resources, to assess the impact of proposed development, and to keep track of land records [Hatch 1986, Parker 1986]. Since 1975, the U.S. Fish and Wildlife Services has been developing a GIS for natural resource applications. They now have predictive techniques that model the effects of channel deepening on the distribution of various shellfish and finfish species [Hatch 1986]. GIS developed by the National Coastal Ecosystem Team have been used to analyze causes and formulate solutions for wetland change/loss [Ader 1982]. The U.S. National Park Services created the Geographic Information Systems Field Unit, charged with developing, applying, and supporting GIS technology throughout the U.S. National Park system [Fleet 1986].

Although GIS technology is being used as a tool for resource management, several areas need improvement. Slow response time is due to the volume of spatial databases required for real world applications. Throughput of a GIS is not always rapid because of the complexity of the geographic data overlay process. Because most GIS are not particularly easy to use, GIS duties are added to existing responsibilities of already burdened personnel [Fleet 1986]. It is unrealistic to expect resource managers to also become experts in GIS. These and other problems of using GIS for resource management are increasingly being addressed by the application of expert system technology. This paper provides an overview on general aspects of GIS technology in resource management, summarizes some research efforts directed towards incorporating

expert systems into GIS, and discusses some prospects for the future application of expert systems to GIS.

Geographic Information Systems

Geographic data describe objects from the real world in terms of their position in a known coordinate system, relationships with one another, and attributes [Burrough 1986]. GIS are designed to manage data gathered by widely disparate methods, integrate the data , provide analysis, and a map product. A GIS should be able to answer a broad range of questions about these data and their relationships.

Data input consists largely of transforming data captured in the form of existing maps, field observations, and sensors into a compatible digital form. The process is one of the most costly and time-consuming elements of GIS development. Existing maps, such as those of vegetation distribution, may be captured by a person working with a digitizer to encode the coordinates of desired point, line, and area objects. Scanners have found a niche in automatic map input, but they are still not intelligent enough to deal with symbolized lines and some of the problems of text recognition [Dangermond and Morehouse 1987]. Aerial photographs still provide much of the information used by resource managers. Capturing environmental data from aerial photographs and encoding them for use in GIS is a complicated task requiring trained personnel. Sensor systems, such as LANDSAT, provide digital image data in a form more readily manipulated by a GIS. Digital map data, like the U.S. Geological Survey's Digital Line Graph or Digital Elevation Model, are becoming common sources of geographic data for GIS.

Geographic data are stored and managed using a variety of structures. Most data structures are based on either a raster (grid cell) or vector (line/polygon) model. The ability of a GIS to integrate raster data from a satellite sensor with vector data digitized from a map sheet is one of its strengths, as well as the source of many problems in data storage and management. Integration of spatial data from a variety of sources means that problems of locational registration must be solved by the GIS so that all data are located on a common frame of reference. This common spatial frame allows the system

to answer queries requiring use of data from a number of different sources. The data integration problem has contributed to making data consistency one of the major problems of GIS [Meier and Llg 1986].

Query operations in a GIS often involve both the retrieval and analysis of geographic data. Once data are retrieved from the database, the query may require that some kind of spatial analysis be conducted. For example, one may wish to determine the best routing for deploying a forest fire fighting unit. This demands the retrieval of possible routes, as well as some spatial analysis to determine the best route.

Query operations are a series of procedures that make up the question-answering process of a GIS. For example, a simple set of query operations would be to open a data file and then select from that data file those records containing the entry 'pine' for the variable tree species. In many of today's GIS, the resource manager would have to specify in detail each of the operations composing a query. Simplifying this interface between GIS and resource managers is an area that requires further development, so that GIS can become more naturally responsive to the resource managers, rather than requiring managers to become experts in GIS operations.

Results of a geographic query can be presented using a range of output methods. However, most users want the graphical output of a GIS to be similar to a conventional map. They require that GIS-produced maps conform to all standards of manual cartography. Thus, even though output techniques of GIS increasingly rely on skills developed in the field of computer graphics, the map remains the most common output of a GIS. Often, users of expensive GIS fail to communicate properly because of a lack of cartographic knowledge on their part or because such knowledge has not been encoded as part of the GIS [Burrough 1986]. This deficiency has led a number of researchers to become interested in the construction of expert systems for map design.

Expert Systems and Geographic Information Systems

Expert systems are computer programs that help solve real-world problems normally requiring an expert interpretation. They are computer models of expert human

reasoning. What areas of GIS operation could benefit from the advice of an 'expert' to solve real-world problems? Some of these areas can be identified by considering some efforts directed at problems related to constructing expert GIS (see Table 1). There are four broad categories into which the efforts fall. They are: geographic feature extraction, geographic database systems, geographic decision support, and map design.

Table 1
Some expert systems for geographic information systems

Problem Domain	Expert Systems	Developers
Geographic feature extraction	ACRONYM	Brooks [1983]
	FES	Goldberg et al. [1984]
	CERBERUS	Engle [1985]
	MAPS	McKeown [1984]
	SPAM	McKeown et al. [1985]
Geographic database systems	ORBI	Pereira et al. [1982]
	LOBSTER	Frank [1984]
	KBGIS	Glick et al. [1985]
	KBGIS-II	Smith et al. [in press]
	SRAS	Robinson and Wong [1987]
Geographic decision support	TS-PROLOG	Barath and Futo [1984]
	URBYS	Tanic [1986]
	GEODEX	Chandra and Goran [1986]
	ASPENEX	Morse [1987]
	OKESS	Lee and Pickford [1987]
	FIRES	Davis et al. [1986]
Map Design	MAPEX	Nickerson and Freeman [1986]
	AUTONAP	Freeman and Ahn [1984]
	MAP-AID	Robinson and Jackson [1985]
	CES	Muller et al. [1986]
	ACES	Pfefferkorn et al. [1985]

Geographic Feature Extraction

ACRONYM is an expert system that was developed to automate image processing and interpretation of aerial photos [Brooks 1983]. ACRONYM incorporates three expert systems, each of which has its own knowledge representation and type of reasoning. All three cooperate to interpret images. The expert systems are: 1) a prediction system using three-dimensional models to predict geometrically invariant features to look for in an image; 2) a description system using an image to obtain descriptions of possible image features; and 3) an interpretation system that uses the descriptions from expert system #2 to find constraints and to check for consistency in the results. Graph matching is used to perform reasoning. The three systems are iterated (prediction to description to interpretation) to get increasingly more detailed interpretations of the image. ACRONYM has been tested on a limited amount of imagery for a small set of objects. Thus, the control mechanisms in ACRONYM are unlikely to be able to handle interpretation of images containing a large set of complex objects [Lambird et al. 1984].

FES (Forestry Expert System) [Goldberg et al. 1984] is used to analyze multi-temporal LANDSAT data for the detection and monitoring of change in forests. It has been trained to analyze forest cover changes in a 50 by 50 km forested region of Newfoundland, Canada. Using a multi-temporal LANDSAT image database, production rules are applied in two phases. First, rules are used to detect change and estimate reliability. They decide whether the changes in classification from one time period to another are real or if they are statistical artifacts. The second phase generates a set of decision rules regarding the current state of the image. For example, a new decision in the second phase could be arrived at in the following manner: "Given that the previous decision was softwood, and a statistically significant change has occurred, specifically softwood to clearcut, it can be concluded that the area represented has undergone logging." This is one of the few expert systems that will explicitly discuss reliability and certainty factors. Very few expert system have attempted to do that.

The control structure of FES is a 'feed forward' system. Once a decision has been reached in the second phase, the system starts another iteration. There is apparently no backtracking, but there is a provision for adding production rules by an expert should an unusual situation occur.

FES was tested using imagery covering 100 km^2 in central Newfoundland, Canada, over a period of six years. Among the test results was the detection of forest regeneration inferred by changes to areas previously classified as clearcut. Spruce budworm damage was inferred from changes in winter imagery that indicated areas previously classified as softwood were appearing as hardwood. They also report an instance where weather phenomena caused a water area to be misclassified as a burn area, and FES, using its accumulated knowledge, rejected this classification and retained the correct, original classification.

Palmer [1984] uses logic programming (PROLOG) as the basis of an expert system for analysis of terrain features. Using a triangular tesselation of elevation data, Palmer represented nodes with their elevation, and segments and triangles as first-order predicates. Then, using PROLOG to conduct symbolic analyses, he demonstrated how valleys, streams, and ridges could be detected using the procedural knowledge encoded in a knowledge base and using PROLOG control mechanisms. This work was elaborated by Frank et al. [1986] who attempted to show how expert system formalism can be used to define uncertain notions about physical geography. They applied first-order predicate calculus in PROLOG syntax to define geomorphology terms.

Geographic Database Systems

ORBI, one of the oldest expert systems implemented in PROLOG, was developed to keep track of the environmental resources of Portugal. ORBI includes features of both a classification system for environmental data and a decision support system for resource planning. ORBI provides graphic input and output of maps via a digitizing table and plotter; a natural language parser for Portuguese that supports pronouns, ellipses, and other transformations; a menu handler for fixed-format input; an explanation facility that keeps track of the steps in a deduction and shows them on request; and help facilities that explain what is in the database, the kinds of deductions that are possible, and the kinds of vocabulary and syntax that may be used.

LOBSTER [Frank 1984] is a query language for a geographic database [Frank 1982]. LOBSTER serves as an intelligent user interface to an object-oriented network spatial database management system. Using the PROLOG syntax to build a user interface to the PANDA database management system [Frank 1984] is an alternative to the extension of standard query languages to include facilities to access GIS. In addition to the intelligent user interface, LOBSTER includes a facility for a programmer to define new 'built-in' predicates that are defined as Pascal programs on the object in the database.

KBGIS is a comprehensive knowledge-based GIS using hybrid knowledge representation [Glick et al. 1985]. Frame-based semantic nets are used to represent the geographical objects and their interrelationships. This representation provides the ability to incorporate new entities, attributes, and relationships. These new entities can also inherit characteristics from their object-types or similar entities. Incorporation of geographical entities is facilitated by the use of an expert system shell that is part of KBGIS. KBGIS has recently become the first operational knowledge GIS system in government and is used on a daily basis for trafficability studies and geopolitical trend analysis.

KBGIS-II [Smith and Pazner 1984] is a prototype knowledge-based GIS designed to speed the search through very large geographic data bases. It uses a spatial object language similar to others based on predicate calculus. Using the spatial object language, a user can edit the knowledge base or perform a query operation. The search procedures are concerned with the satisfaction of spatial constraints, primarily whether an object lies within a particular window (e.g., rectangular area). In the database, spatial objects are represented in a specialized hierarchical grid structure called a quad-tree. KBGIS-II also uses some primitive learning procedures to make spatial searches more efficient. Hardware deficiencies do not permit this system to be truly interactive [Smith et al., in press].

SRAS [Robinson and Wong 1987] is a Spatial Relations Acquisition Station designed to acquire representations of spatial relations expressed as natural language concepts (such as near, far, close to, remote from) for use in subsequent queries of a geographic database. It addresses the problem of how to represent natural language

concepts so that they can be used to answer geographic queries. For example, a resource manager may want to know which streams are 'near' a proposed waste disposal site. SRAS would interact with the user to arrive at an explicit representation of the user's meaning of 'near'.

SRAS is based on a process for acquiring linguistic concepts through mixed-initiative human-machine interactions. The process, using fuzzy topological sets, is designed to determine the meaning of a spatial relation such as 'near'. First, the system is initialized at a maximum uncertainty condition. Second, SRAS chooses a question so that a measure of expected certainty is minimized. Third, the user's response to the question is used to adjust representation of the linguistic variable. Finally, before starting question selection and concept processing again, a process controller determines whether the fuzziness of the concept has been reduced to an acceptable degree of approximation. This process is unique because it uses fuzzy representations while only requiring the user to answer the questions with a simple yes or no.

SRAS can also be used to acquire multiperson concepts. This ability has particular relevance to the usual organizational context of GIS, where terms and relations are the product of committee definition or review [Robinson and Wong 1987]. This station has potential value for others seeking to develop systems for acquiring spatial knowledge for use in expert systems.

Geographic Decision Support

ASPENEX [Morse 1987] is an expert system to aid management of aspen resources in the Nicolet National Forest, Wisconsin. ASPENEX interfaces an expert system with a public domain GIS, MOSS (Map Overlay and Statistical System). The user interface, rule base, and interprogram communication were developed using the shell, EXSYS, on a microcomputer. The microcomputer resident expert system provides rules required in aspen management and MOSS provides spatial information on characteristics of aspen stands.

Barath and Futo [1984] developed a logic-based expert system for comparing requirements of economic sectors (e.g., agriculture, industry, construction) and social factors (e.g., living conditions, employment, education, culture) for making planning decisions. The goals of a district are evaluated in light of the planning actions to be taken, such as building a school, and the of time and cost. Alternative actions may also be evaluated. For example, the system evaluates the feasibility of building a road and a school or just a school.

The system is based on a hierarchical model of national, county, and district-level activities. This goal-directed system was first developed in T-PROLOG and is now based on the TS-PROLOG language. TS-PROLOG is a version of standard PROLOG [Clocksin and Mellish 1981] that has been extended to allow for parallel processes and system simulation. Processes behavior is described by Horn clauses. Therefore, extensions to PROLOG are kept consistent with the representational calculus of logic programming. This system is still an experimental application, primarily funded by the Ministry of Industry of Hungary.

GEODEX [Chandra and Goran 1986] was built to help planners evaluate site suitability for particular land-use activities. Its rules were drawn from an expert land-use planner. Using the rules in its knowledge base, GEODEX uses a constraint-directed search strategy. For example, when locating a shopping centre, constraints such as nearness to a primary road and distance from schools are used to determined the suitability of a site. The land-use planner can develop the complete bundle of constraints, in the form of rules, that apply to the location of a particular land use.

Although the Map Analysis Package (MAP) [Tomlin 1983], a well-known FORTRAN package for geographic information processing of raster data, was used to perform map overlays, GEODEX (written primarily in LISP) does not interface with any geographic information system. It represents maps in an object-oriented format with each object having a frame of associated information. The frame information is extracted and used by the rule-based system. The map overlays performed by MAP are used to provide information for the frames.

OKESS [Lee and Pickford. 1987) is a prototype expert advisory system for use in dispatching suppression forces to wildfires on the Okanogan National Forest, Washington. It is a PROLOG based expert system using decision trees as the initial knowledge base. It uses time of year and a variety of data about burning conditions, characteristics of the area in which the fire is burning, past and projected weather, smoke dispersal, etc., to arrive at a decision as to whether to immediately suppress the fire, to contain the fire within set boundaries, or to confine it within a predetermined drainage or area.

Map Design

The MAP-AID expert system being developed in the United Kingdom [Robinson and Jackson 1985] is an attempt to develop a general purpose expert cartographic system. The goal is to improve the quality of maps derived from GIS by taking complete control of map design and production. MAP-AID is divided into four components: 1) the 'core', containing the map design rule base and other information held as rules in the knowledge base, 2) the user module through which the user controls the system and interacts with the knowledge base, 3) a set of data-system modules, and 4) a set of graphics package modules. PROLOG is being used as the primary language for development of this expert system, but the project has faltered due to unanticipated difficulties in formalizing cartographic knowledge.

CES is a prototype cartographic expert system intended for use by Energy Mines and Resources as an advisor to help cartographers design the Electronic Atlas of Canada. Using decision tables, CES provides a set of mapping specifications that optimally fulfill a set of map requirements given to the system. Preliminary responses from user indicate that cartographers disagree 1) on taxonomy of cartographic concepts that should be introduced in the system, 2) on the rules associating map requirements to map specifications, and 3) on the formal mapping requirements derived from their specific needs. This project experienced some of the same problems as MAP-AID when its developers discovered that cartographic knowledge is difficult to formalize and sometimes inconsistent [Muller 1986, Muller et al. 1986].

AUTONAP [Ahn 1984, Freeman and Ahn 1984] is one of the most successful name placement expert systems developed so far. This system emulates an expert cartographer in the task of placing feature names on a geographic map. It uses heuristic knowledge about map placement based on established procedures and conventions. The knowledge base consists of a small set of about 30 explicit rules organized as subroutines in a large RATFOR program on a Prime 750. In AUTONAP, area features are annotated first, then point features, and finally line features. In this manner the system progresses from the most constrained to the least constrained feature annotation task. Some backtracking is part of the system.

MAPEX is a rule-based system that emulates a cartographer in the task of defining and maintaining the high-level decisions of map generalization. English-like rules are compiled into a FORTRAN-77 subroutine. To facilitate generation of a formal grammar, generalization actions are described according to data types such as hydrography, transportation, or cultural features. The data types used in the formal grammar of rule building are dictated by conventions used by the U.S. Geological Survey's Digital Line Graph [Allader and Elassal 1983]. MAPEX splits automated map generalization tasks into high-level rule definition and translation and low-level geometric manipulation of map data.

Prospects

Some of the most promising applications of expert systems for GIS lie in the development of intelligent user interfaces, efficient spatial database search techniques, improved image classification, and production of high quality cartographic products [Ripple and Ulshoefer, in press]. There are many other areas suitable for expert system development within each of the major components of GIS—input, management, query, and output—where decisions must be made that will affect both the efficiency and effectiveness of the GIS.

Geographic Data Input

Geographic data input encompasses all aspects of transforming data captured in the form of existing maps, field observations, and sensors into a compatible digital form.

Scanners are being used to transform manuscript maps into digital representations. Contrary to earlier predictions, scanners have not taken over the capture of cartographic data. Dangermond and Morehouse [1987] explained that a lot of map editing is going on at the same time as manuscripts are being captured. For expert systems to automatically capture manuscript maps, they must have knowledge of the scanning system, map object modeling, image analysis, and the map revision process.

Field observations range from the precise measurements of the surveyor to the field notes of a wildlife biologist. There is a need for expert systems to translate these observations, some in digital form and some in descriptive language, into a form useful within a GIS. These raw data must be organized and subsequently checked for consistency. The input and transformation of field notes by an expert system would demand a sophisticated understanding of the problem domain language. Of substantial importance would be the ability of the system to understand spatial descriptions at a level permitting the inference of a location in coordinate space.

Lambird et al. [1984] argue that past techniques for the automated or semi-automated extraction of geographic features from remotely sensed images did not work well because of 1) the large amount of information in each image, 2) ambiguous and contradictory information present in the images, 3) problems of dealing with perspective changes, object scale, and resolution, 4) many sources of geometric and radiometric variability, and 5) lack of adequate models relating physical principles to object appearance in images. The advent of expert systems may solve many of these problems. Expert systems that interpret images require knowledge about image processing, image formation, object recognition and modeling, and the ability to develop some sort of certainty measure and competent factor. This knowledge is generally distributed among several experts. Expert systems like ACRONYM may exploit the advantages of distributed problem solving.

There is an increasing amount of data that is not of image form but is best characterized as digital-spatial data. The spatial data provided in digital form may not contain the features a resource manager finds useful, but may contain the data from which those features must be inferred. For example, much resource management is oriented around stream networks and watersheds. Band [1986] has shown one way

that stream networks and their accompanying watersheds can be extracted from a digital elevation model. However, the extraction of high-level geographic features requires not only knowledge about their geometry, but also domain specific expertise such as that possessed by physical geographers, ecologists, and foresters.

Geographic Data Storage and Management

Although systems such as URBYS are using data bases, it is clear from the systems discussed above that the methods of storing geographic data are changing in response to the demands of expert systems. In an effort to develop storage structures that support rapid spatial searches, some have begun to base their systems upon the quad-tree [Antony and Emmerman 1986; Smith et al., in press]. Regardless of the specific storage model, several areas of geographic data management require significant expertise.

The ability of a GIS to integrate a number of data layers is one of its strengths. However, this is also the source of many problems. Expert systems developed to manage the geographic database will need knowledge of positional modeling, spatial interpolation, geographic consistency, and uncertainty modeling. The geographic database manager needs to be able to assess the uncertainty characteristics of incoming data and take appropriate action. Many experts in GIS have commented that some people are adept at increasing the certainty of resource data using a reasoning process whereby they apply their expertise and knowledge about ancillary data to arrive at an improved data layer. (An example might be where there is uncertain data about tree species in an area with mountains and valleys. Since some species tend to be located at particular elevations on the windward side of a mountain, use of elevation and location on the mountainside can help improve the certainty with which we can make statements about the distribution of tree species.)

It is, however, important that the various data layers be consistent with one another. Otherwise, logical inconsistencies may arise in subsequent analyses. Meier and Llg [1986] have discussed some general rules of consistency in a simple spatial database.

Much remains to be done to formalize the rules needed for maintaining consistency in a complex, multi-layered geographic database.

One of the keys to the development of expert geographic database managers is managing the different kinds of uncertainty found in geographic data bases [Robinson and Frank 1985]. Some of developments in this area are seen in the inclusion of a reliability measure in FES, Shine's [1985] review of the utility of Bayesian, fuzzy, and belief logics in feature extraction systems, and Robinson's [in press] discussion of the implications of fuzzy logic for geographic data bases.

Geographic Query

Both the retrieval and analysis of geographic data are often part of a geographic query. Retrieval of geographic data demands that some kind of spatial search be conducted. Increasing the efficiency of spatial search is one of the major objectives of KBGIS-II. However, due to reported hardware limitations, KBGIS-II is not truly interactive [Smith et al., in press].

An intelligent user interface could guide an inexperienced user through the most efficient use of the system [Jackson and Mason 1986], but the interface would have to be an expert on that particular GIS. Resource managers, who have little time to become experts in GIS as well as in resource management, would welcome such an interface. An intelligent user interface is one of the concepts behind the development of ASPENEX [Morse 1987].

One of the problems in developing an expert system interface is to provide a natural means of communication between the manager and the GIS. This communication demands knowledge of what is meant by the manager's request. Thus, semantic modeling of resource management queries is needed to develop formal knowledge of how to interpret queries. Robinson and Wong [1987] have shown how seemingly trivial queries can vary, depending upon the user. In controlled, experimental sessions with SRAS, none of five subjects agreed on the meaning of 'far from Douglas, Georgia'. In fact, a user with graduate-level training in computer cartography gave

answers that resulted in an inconsistent meaning of 'close to Douglas'. In other words, the user said one settlement was close to Douglas, which is farther away than another settlement which was said to be not close to Douglas. Furthermore, out of five subjects, eight terms used to define relations, and two data bases, there was, on the average, less than 50 % agreement on the meaning of terms describing spatial relations.

Map Design

The map remains one of the most common output products of a GIS. However, users of expensive GIS sometimes fail to communicate properly because of a lack of cartographic knowledge [Burrough 1986]. MAP-AID and CES both encountered problems due to the informal nature of current cartographic knowledge. The formalization of cartographic knowledge remains one of the challenges of constructing cartographic expert systems [Fisher and Mackaness 1987]. As discussed above, progress is being made in both automatic name placement and map generalization.

Another aspect of the map design process is identification and resolution of spatial conflicts when placing symbols on a map. Mackaness and Fisher [1987] have attempted to formalize the process of automatic recognition and resolution of spatial conflict in cartographic symbolizations, but further research is required to resolve some of the problems of overlap, enclosure, and worst case resolution that were identified by their investigations. Like others working on cartographic expert systems, their goal is to enable researchers with no cartographic skills to display field data using designs that best communicate those data.

Concluding Discussion

Resource managers make complex planning and management decisions daily. These experts are scarce and in high demands. Expert Systems, in general, can provide a more permanent, accessible, and consistent source of expertise [Stock 1987].

Several areas in the application of expert systems to GIS still need to be improved. One of them is natural language interface, from the points of view of both geographic

data input and query. It is difficult to determine in an accurate and consistent manner what a spatial relationship is in an expression or in a sentence. Fuzzy set theory and spatial linguistics will be the important issues to the development of GIS and other applications in autonomous vehicles.

Another area that received less attention is in the reasoning or explanatory capability of expert systems, which is still in a very primitive state of development [Stock 1987]. The self-knowledge or explanatory capability of cartographic expert systems is presently critical to their successful use because automated methods of deciding among competing cartographic designs have yet to be developed [Fisher and Mackaness 1987]. The formalization of knowledge of this type needs to receive substantial emphasis in future development of expert GIS.

Development of smaller prototypes using structured knowledge acquisition methodologies and semantics will not only bring practical results, but, more importantly, will help clarify many less formalized areas. Lessons learned in building these smaller systems will in turn be transportable and expandable to later, more advanced systems. By developing more than one prototype addressing the same problem but using different methods, we can begin to conduct comparative analyses that can support more informed decisions concerning the development of expert systems for GIS.

Acknowledgements

This research received partial support from the Alberta Forestry, Lands, and Wildlife Professorship in Digital Mapping and Spatial Data Management and a grant from the Natural Sciences and Engineering Council of Canada.

References

Ader, R.R. (1982). A geographic information system for addressing issues in the coastal zone. *Computers, Environment, and Urban System* 7(4):233-244.

Ahn, J.K. (1984). Automatic map name placement. IPLTR-063, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York.

Allader, W.R., and A.A. Elassal (1983). Digital line graphs from 1:24,000-scale maps. United States Geological Survey Circular 895-C, United States Geological Survey, Reston, Virginia.

Antony, R., and P.J. Emmerman (1986). Spatial reasoning and knowledge representation. In: *Geographic Information Systems in Government*, ed. B. Opitz. A. Deepak Publishing, Hampton, Virginia, pp. 795-814.

Band, L.E. (1986). Topographic partition of watersheds using digital elevation models. *Water Resources Research* 22: 15-24.

Barath, E., and I. Futo (1984). A regional planning system based on artificial intelligence concepts. *Papers and Proceedings of the Regional Science Association* 55: 135-154.

Brooks, R.A. (1983). Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-5: 140-150.

Burrough, P.A. (1986). *Principles of Geographic Information Systems for Land Resources Assessment.* Oxford University Press, New York.

Chandra, N., and W. Goran (1986). Steps toward a knowledge-based geographical data analysis system. In: *Geographic Information Systems in Government,* ed. B. Opitz. A. Deepak Publishing, Hampton, Virginia, pp. 749-764.

Clocksin, W., and C. Mellish (1981). *Programming in PROLOG.* Springer-Verlag. New York.

Dangermond, J., and S. Morehouse (1987). Trends in hardware for geographic information systems. *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 380-385.

Davis, J.R., J.R.L. Hoare, and P.M. Nanninga (1986). Developing a fire management expert system for Kakadu National Part, Australia. *Journal of Environmental Management*, 22: 215-227.

Engle, S.W. (1985). Cerberus release notes, version 1.0. NASA Ames Research Center, Moffet Field, Ca.

Fisher, P., and W. Mackaness (1987). Are cartographic expert systems possible? *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 530-534.

Fleet, H. (1986). Geographic information systems and remote sensing activities in the National Park Service. In: *Geographic Information Systems in Government,* ed. B. Opitz. A. Deepak Publishing, Hampton, Virginia, pp. 635-644.

Frank, A.U. (1982). MAPQUERY: Data base query language for retrieval of geometric data and their graphical representation. *Computer Graphics* 16: 199-207.

Frank, A.U. (1984). Extending a network database with PROLOG. *Proceedings, First International Workshop on Expert Database Systems*, Kiawah Island, South Carolina, October, pp. 665-674.

Frank, A.U., B. Palmer, and V.B. Robinson (1986). Formal methods for accurate definitions of some fundamental terms in physical geography. *Proceedings, Second International Symposium on Spatial Data Handling*, Seattle, Washington, July, pp. 583-599.

Freeman, H., and J. Ahn (1984). AUTONAP—An expert system for automatic map name placement. *Proceedings, First International Symposium on Spatial Data Handling*, Zurich, Switzerland, August, pp. 544-571.

Glick, B., S.A. Hirsch, and N.A. Mandico (1985). Hybrid knowledge representation for a geographic information system. Presented at the Seventh International Symposium on Automated Cartography, Ottawa, Ontario, March.

Goldberg, M., M. Alvo, and G. Karam (1984). The analysis of LANDSAT imagery using an expert system: Forestry applications. *Proceedings, Sixth International Symposium on Automated Cartography*, Ottawa, Ontario, March, pp. 493-503.

Hatch, H.J. (1986). Overview of geographic information systems in the federal government. Geographic Information Systems in Government, ed. B. Opitz. A. Deepak Publishing, Hampton, Virginia, pp. xvii-xx.

Jackson, M.J., and D.C. Mason (1986). The development of integrated geo-information systems. *International Journal of Remote Sensing* 7: 723-740.

Jordan, G., and L. Vietinghoff (1987). Fighting spruce budworm with a GIS. *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 492-499.

Lambird, B.A., D. Lavine, and L.N. Kanal (1984). Distributed architecture and parallel non-directional search for knowledge-based cartographic feature extraction systems. *International Journal of Man-Machine Studies* 20: 107-120.

Lee, B.S., S.G. Pickford (1987). An Expert System for use in dispatching suppression resources to wildfires. Presented at the 9th Conference on Fire and Forest Meteorology, San Diego, California, April 22-24.

Mackaness, W.A., and P.F. Fisher (1987). Automatic recognition and resolution of spatial conflicts in cartographic symbolization. *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 709-718.

McKeown, D.M. (1984). Digital cartography and photointerpretation from a database viewpoint. In: *New Applications of Databases*, eds. G. Gargarin and E. Colembe. Academic Press, New York, pp. 19-42.

McKeown, D.M., W.A. Harvey, and J. McDermott (1985). Rule-based interpretation of aerial imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7: 570-585.

Meier, A., and H. Llg (1986). Consistent operations on a spatial data structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8: 532-542.

Morse, B. (1987). Expert interface to a geographic information system. *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 535-541.

Muller, J.C. (1986). Personal communication. University of Alberta, Edmonton, Alberta, November.

Muller, J.C., R.D. Johnson, and L.R. Vanzella (1986). A knowledge-based approach for developing cartographic expertise. *Proceedings, Second International Symposium on Spatial Data Handling*, Seattle, Washington, July, pp. 557-571.

Nickerson, B.G., and H. Freeman (1986). Development of a rule-based system for automatic map generalization. *Proceedings, Second International Symposium on Spatial Data Handling*, Seattle, Washington, July, pp. 537-556.

Nilsson, N.J. (1971). *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York.

Palmer, B. (1984). Symbolic feature analysis and expert systems. *Proceedings, First International Symposium on Spatial Data Handling*, Zurich, Switzerland, August, pp. 465-478.

Parker, H.D. (1986). Geographic information systems technology in natural resource management: the Bureau of Land Management's example. In: *Geographic Information Systems in Government*, ed. B. Opitz. A. Deepak Publishing, Hampton, Virginia, pp. 1-6.

Pereira, L.M., P. Sabatier, and E. de Oliveira (1982). ORBI—An expert system for environmental resource evaluation through natural language. Report FCT/DI-3/82, Departmento de informatica, Universidade Nove de Lisboa.

Pfefferkorn, C., D. Burr, D. Harrison, B. Heckman, C. Oresky, and J. Rothermel. (1985). ACES: A cartographic expert system. *Proceedings, Seventh International Symposium on Automated Cartography*, Washington, D.C., March, pp. 399-407.

Ripple, W.J., and V.S. Ulshoefer (In Press). Expert systems and spatial data models for efficient geographic data handling. *Photogrammetric Engineering and Remote Sensing* 53(10).

Robinson, V.B. (In Press). Some implications of fuzzy set theory applied to geographic databases. *Computers, Environment, and Urban Systems.*

Robinson, G., and M. Jackson (1985). Expert systems in map design. *Proceedings, Seventh International Symposium on Automated Cartography*, Washington, D.C., March, pp. 430-439.

Robinson, V.B., and A.U. Frank (1985). About different kinds of uncertainty in geographic information systems. *Proceedings, Seventh International Symposium on Automated Cartography*, Washington, D.C., March, pp. 440-450.

Robinson, V.B., and R.N. Wong (1987). Acquiring approximate representations of some spatial relations. *Proceedings, Eighth International Symposium on Automated Cartography*, Baltimore, Maryland, March, pp. 604-622.

Shine, J.A. (1985). Bayesian, fuzzy, belief: Which logic works best? *Proceedings, American Society of Photogrammetry*, Washington, D.C., March, pp. 676-679.

Smith, T.R., D. Peuquet, and S. Menon (In Press). KBGIS-II: A knowledge-based geographic information system. *International Journal of Geographic Information Systems.*

Smith, T.R., and M. Pazner (1984). Knowledge-based control of search and learning in a large-scale GIS. *Proceedings, First International Symposium on Spatial Data Handling*, Zurich, Switzerland, August, pp. 498-529.

Stock, M. (1987). AI and expert systems: An overview. *AI Applications in Natural Resource Management*, Vol. 1, No. 1, pp. 9-17.

Tanic, E. (1986). Urban planning and artificial intelligence; The URBYS system. *Computers, Environment, and Urban Systems* 10(3-4): 135-146.

Tomlin, D. (1983). Digital cartographic modeling techniques in environmental planning. Ph.D. dissertation, Yale University, New Haven, Connecticut.

Webb, P.R. (1982). A synopsis of natural resource management and environmental assessment techniques using geographic information system technology. *Computers, Environment, and Urban Systems* 7(4): 219-232.

TOWARDS A SPATIAL THEORY

Andrew U. Frank
Surveying Engineering Program
University of Maine
Orono, ME 04469
U.S.A.
FRANK@MECAN1.bitnet

Abstract

In several recent publications, problems with spatial reasoning and the lack of a coherent spatial theory have been noted. Human beings are very good at analyzing spatial situations, determining patterns and similarities, etc.; however, programmed systems on computers cannot compete.

The well-known Euclidian geometry is not a true image of how human beings think about geometry. The ideal objects of Euclidian geometry are quite unlike the real, extended objects of which the world is made. Moreover, Euclidian geometry cannot be easily modelled on computer systems. The absence of a coherent spatial theory hinders development of GIS. Building a comprehensive, coherent spatial theory will not be simple. Several different approaches are currently tried: (1) defining algebras with finite representation that can model Euclidian geometry, (2) using combinatorial topology to represent geometry with simplices, and (3) applying models based on finite resolution rasters.

It is important that multiple approaches are tried and ultimately combined to best accommodate the concepts human beings use. In cognitive science, efforts to better understand spatial concepts will be contributed to a spatial theory.

1. Introduction

The development of computer systems to the present level of performance and reduced cost make many applications possible which were not conceivable a number of years

ago. Computers can now be used to perform operations that in the past were carried out tediously by hand. We can also see new applications which were not possible with the traditional technology. This development does not only influence the technical solutions we select or the problems we can successfully attack, but it will also influence the scientific concepts we use [Cebrian de Miguel and Sendra 1987]. This paper deals in the foreground with practical problems and possible technical solutions, but the methods proposed can be used to further the theoretical understanding of spatial situations and processes.

Spatial information systems, used as the common name for geographic information systems and land information systems [FIG 1981], are in a way a combination of traditional tasks and new concepts. Several decades ago, the possibility for this development became visible and a large number of optimistic predictions have been written; however, an assessment today reveals that only a few systems are productive and their use, although beneficial, is much less general than thought possible.

The development of software for spatial information systems falls behind user demands and some of the available programs still show major shortcomings. For example, none of the commercially available GIS software systems contain an algorithm for polygon overlay which is absolutely foolproof. The best ones work in practical cases, but some fail even with relatively simple, practical problems. A recent discussion with a development team confirmed that the programming of geometric algorithms is the major difficulty. It was generally believed that the geometric operations were simple to program — after all, "this is (two-dimensional) high school analytical geometry". Unfortunately, this is not correct and we begin to understand our errors. We still do not completely understand the tasks that human beings and computers do well. Presently, human reasoning exceeds the computer's ability to analyse and deal with spatial situations. This paper attempts to outline a research program to understand the human capabilities better, and find possibilities to model them in computer systems.

Too often computer geometry and computer graphics have been lumped together. In this report we separate the concerns of processing geometric data from considerations for output presentation. Clearly, graphical presentation methods are based on geometric

processing, but may have different requirements than processing of geographic data for analytical purposes.

As a second level of separation, we advocate the analysis of geometric aspects of information in a separate step from the treatment of the associated attribute data. Clearly a spatial information system must contain routines to treat both classes of data, but a separation of concerns will simplify the design and coding [Frank 1987]. This paper deals with the spatial aspects only.

2. The Concept of Spatial Theory

Boyle [1983] reports on a NASA sponsored conference about the "review and synthesis of problems and directions for large scale cartographic information system development." The conference report contains an extensive discussion of the need for a spatial theory — a term which was first used by Chrisman [1975]:

> There is at present no coherent mathematical theory of spatial relations.
>
> The world in which we live is a spatial world and to exist in it all of us master spatial relations on an intuitive basis. While these relationships are highly complex, we usually need not even think about them. We make use of the eye/brain combination to sort out the visual images we receive and recognize their various elements.
>
> In dealing with spatial data in the form of maps, images and the like, we usually obtain the data with our eyes and then deal with spatial aspects of the data on an intuitive basis.
>
> Unfortunately, when a computer must be instructed in how to perform such operations, intuition must be replaced by precise statements and by precise mathematical relationships.
>
> At present we cannot supply this need.
>
> The lack of a coherent theory of spatial relations hinders the use of automated geographic information systems at nearly every point. It is difficult to design efficient data bases. Difficult to phrase queries of such data bases in an effective way, difficult to interconnect the various subsystems in ways which enhance overall system function, difficult to design data processing algorithms which are effective and efficient. As we begin work with very large spatial data bases or global data bases the inabilities and inefficiencies which result from this lack of theory are likely to grow geometrically.

> While we can continue to make some improvement in the use of automated geographic information systems without such a coherent theory on which to base our progress, it will mean that the development will rest on an inevitably shaky base and progress is likely to be much slower than it might be if we had a theory to direct our steps. It may be that some advances will simply be impossible in the absence of a guiding theory. [Boyle 1983]

A spatial theory must therefore include the following points:

- Define a set of generic geometric objects and generic operations on them. Special attention must be directed to include special cases and error conditions.
- The generic objects and operations must be suitable for modelling the geometric properties in a spatial information system; that is, the higher level objects are distinguished by their attributes which restrict and redefine the geometric operations applicable. The quality of the theory is judged by its capability to reduce redundancy and possible inconsistencies in the definition of specific geometric objects.
- Explain the behavior of commonly used operations on geometric objects in terms of the previously precisely defined objects and operations. The success of these definitions can be measured if the commonly used operations can be described easily.
- A spatial theory should be as general possible, thus ensuring a wide range of applications. In the following proposal, we seek for a theory which is independent of the dimensions of the objects it is applied to (i.e., it can be used for the two- or the three-dimensional case).

2.1 Need For A Spatial Theory

A coherent spatial theory should tie together the different concepts of space which are used by the builders and users of a spatial information system. The differences in concepts between the mathematically inclined, computational geometry research and the practical users of systems are wide, and the gap is difficult to bridge. The differences appear in the user interface where users are surprised by system reactions because the concepts they use and the spatial concepts embedded in the system differ substantially. Building comfortable interfaces is only possible if a clear understanding of the user concepts is available, or a set of coherent concepts, which are used for the design of the interface, can be taught to the user. In order to have the system perform the operations

on the user level — and such operations may be different for different classes of applications — such concepts must be general enough that they can be used internally for the design of the basic geometric software. The development of geometric software is so difficult and thus expensive, that a generalized solution can be developed which is useful for spatial information systems as well as for all physical CAD/CAM systems for mechanical and civil engineering.

Clear definitions for spatial concepts are also crucial for data exchange. Transfer of data from one system to another is difficult if the model used for spatial representation differs significantly. A standard for exchange of spatial data needs a formal definition of spatial concepts which can be used to establish conversion routines.

2.2 What Spatial Theories Are Available

A number of theories deal with space from different points of view. We often revert to the high school form of Euclidian geometry and very few know of "elementary geometry from an advanced standpoint" [Moise 1963]. Mathematicians have further studied topology, including graph theory (which is often used in spatial information systems). Analytical geometry shows how to transform geometric objects and operations to the realm of computations (and computers).

Geodesy deals with some aspects of physical space and how they can be measured precisely, whereas GIS often uses rasterized methods to approximate the spatial distribution of some properties. Physics, in general, includes some aspects of the geometric properties of objects — the recent development of "naive physics" [Hobbs and Moore 1985] attempts to capture some of the common understanding of geometric properties of reality with less abstraction than is evident in the pure theory.

3. Classical Geometry vs. 'Naive Geometry'

Geometry deals with abstract objects, like infinitely small points, lines which extend into infinity, etc. Physical objects in the real world are very different: points have a

finite size, lines are finite and never truly straight or parallel. Moreover, our abilities to measure are limited and only approximations for the true values can be determined.

In this section, first the concept of geometry are discussed as they are seen by classical and modern mathematics before extending it to take into account the restrictions of real physical objects.

3.1 Classical View of Geometry

Classical geometry, as developed by the Greeks and usually taught in high school, deals with abstract objects (primarily points and lines) and their interaction. All operations are restricted to be performed, at least in principle, by compass and straight edge.

These abstractions have been very helpful for a simple axiomatization of geometry, but it must be remembered that there are no objects in reality like points and lines, or relations like parallel.

3.2 Concept of Geometry

Geometry in modern mathematics is broader than Euclidian geometry. The aim is to determine the abstract concept of geometry. Since the work of Felix Klein [Klein 1872] it has been accepted that geometry deals with invariances of figures under transformations. This notion seems to capture more of what humans understand by "geometry" than the classical operations with points and lines. It is evident that not only Figures 1a and 1b are related, but most will also see a relation between Figures 2a and 2b. The figures 1 are related by similarity transformations (translations, rotations and scaling), whereas the figures 2 are related by more general topological transformation.

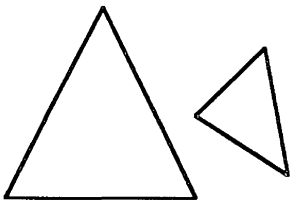


Figure 1 (a) and (b)

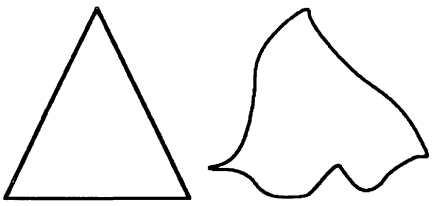


Figure 2 (a) and (b)

In a spatial information system we often want to search for similar patterns, but do not expect to find the same geometric figure again. A number of geographical studies, for example migration analysis of geomorphology, try to find similar spatial patterns. It is an important task for a spatial theory to explain what 'similar' means in each of these cases. There have been attempts to define patterns by using symbolic processing based on a triangular surface [Frank et al. 1986]. The class of continuous transformations in mathematical geometry is too broad and limited. Sometimes, we judge two figures, which are linked by a continuous transformation, as quite different (Figures 3a and 3b), or we judge two figures as similar, even if they are topologically different (Figures 4a and 4b) [Walker 1987].

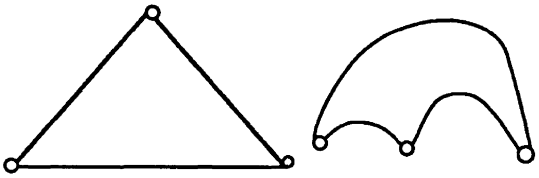


Figure 3 (a) and (b)

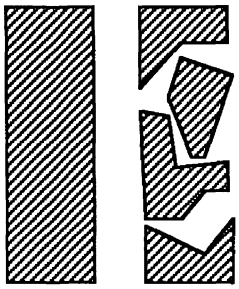


Figure 4 (a) and (b)

3.3 Naive Geometry

The objects we deal with daily are quite different from the abstract geometric concepts we encounter in the mathematical treatment of geometry. The extension of physical objects is limited, but most geometric concepts (e.g., line) are infinite. Secondly, geometry is based on the ability to measure without error, but all our measuring methods are inherently imprecise to a certain degree.

Our daily operations are not severely limited by these problems. As human beings we use a sort of 'naive' geometric reasoning, taking into account automatically the differences between abstract geometry and the geometric properties of real objects. Despite the fact that we all know how to do this, there is no formal understanding of this

process. A formal description of this 'naive' reasoning is necessary in order to have computers simulate such behavior. It seems worthwhile to study if 'fuzzy logic' [Zadeh 1981] applied to our reasoning about spatial relations ('close,' 'next to,' 'far,' etc.).

In the arena of expert system construction, a similar disagreement was found between the laws of theoretical physics (i.e,. what was learned in college in Physics I and II) and the concept humans apply in daily life — we take into account all the impurities of friction in motion without much trouble. The attempts to formalize this 'naive physics,' so as to be incorporated into an expert system has been difficult, but important in areas like building expert systems to assist in the diagnosis of faults in mechanical systems (e.g., railway engines).

A spatial theory should include a 'naive geometry' in the above sense. It must be expected, that this would also be helpful for the development of the 'naive physics,' because much of physical reasoning is based on geometry.

4. Mathematical Geometry vs. Modelling Power of Computer Systems

Modelling of geometry in computers is most always attempted based on analytical geometry. Analytical geometry provides a convenient base for translating geometric problems into numerical ones, thus making them more suitable for computer processing.

Unfortunately, the use of analytical geometry for computer modelling of geometric figures has some serious shortcomings. Analytical geometry needs a mapping to real numbers. The finite approximations in a computer system are not sufficient. The floating point numbers (REAL) of a FORTRAN program are much more akin to the integers: there is only a finite number of them, and between some of the floating point numbers there are no others.

Mathematically speaking, there is a difference between the field of real numbers and the integral domain (a ring). The Euclidian concepts of geometric space are intrinsically linked to the properties of spaces built over real numbers. It was shown that analytical

geometry using computer approximations is not in every case invariant under translation and rotation — the minimal requirements in Euclidian geometry [Franklin 1984].

5. Raster Based Systems vs. Vector Systems

Many useful spatial information systems are not based on a geometric description of objects with vectors, but use a regular subdivision of space in a raster and then record the interesting properties for each raster cell.

Raster systems are based on an intrinsically different concept of space than a vector system. In a vector system, space is the total of the extension of all the objects, each with its own identity, special form, and properties, and the space is organized as a function of these objects. In a raster system, space is an abstract concept which is subdivided in abstract cells, for which we know some properties.

In a raster system, objects can be approximated by a set of raster cells, but a raster system can not deal with the abstract geometric objects like points and lines, and must approximate them with small areas. Typically, line oriented operations are not possible, but a number of other interesting operations (shrink and grow) are available. This is primarily studied in image processing literature. In Serra [1982] an algebraic treatment of typical raster operations (transformations) is proposed.

6. Spatial Relationships

In general we are not only interested in spatial objects, but also in the spatial relations between them. Again, we lack a comprehensive, formally defined set of spatial relationships.

6.1 Distance and Direction

In Euclidian geometry (generally in a metric space) there is a distance defined between any two points with the ordinary properties:

- $d\,(a,a) = 0$

- $d\,(a,b) = d\,(b,a)$
- $d\,(a,b) = d\,(b,c) \geq d\,(a,c)$

In an Euclidian space (generally in any vector space) there is also a direction defined.

Both of these operations are defined between two points. It is interesting to see that distance and direction are not defined in the same type of space and are thus not invariant under the same transformations; however, both obey linear transformations to be made compatible with rotation or scaling.

Human beings use distance daily between extended areal objects to discuss the direction between two areas (Canada is north of Maine? — it is also to the east and to the west). It seems possible to treat distance and direction between extended objects using intervals and then apply interval relations which are closely related to the inclusion relations between extended objects. An obvious first idea is to replace the two objects by their 'center' (e.g., their center of gravity) and then determine distance and direction between these. This does not adequately represent human beings' concept in a large number of cases. Peuquet proposes a method based on heuristics to deal with distance and direction between extended objects, which avoids the most obvious shortcoming [Peuquet and Zhan Ci-Xiang 1987]. We experiment with a system based on an algebra of intervals.

6.2 Boolean Operations

For objects in a topological space, an intersection operation is derived from set intersection. It is possible to determine if an object is spatially included in another, if they overlap or are disjoint. It is also possible to define multiple kinds of 'touching' between objects; moreover, a complete formalization has not yet been achieved.

6.3 Natural Language Concepts for Spatial Relationships

Natural languages use a number of concepts to express the relationships between objects in space. Investigations are just beginning, which analyse the concepts and separate methods, specific to some languages, and others which seem universal [Mark et al. 1987].

In natural language, human beings use concepts which are not precisely defined, like 'far' and 'near'. Their interpretation clearly depends on the context of their usage (e.g., 'near' applied to a gas station is different from 'near' applied to a major airport), but are not intended to be precise even in a single context. It will be difficult to capture these effects and model them. Fuzzy logic [Zadeh 1981] was used to observe the usage of 'near' and 'far' in a set context [Robinson and Wong 1987].

We can assume that the concepts used to express spatial relationships in natural languages are the 'natural' concepts for human beings. It is important to understand them, formally define them and use them in at least two contexts: as key words in query languages and in a system where the human being receives guidance to find a location from an information system.

6.3.1 Query Languages

Useful GIS must provide an interactive facility to ask questions to the system and see immediately answers in the form of small maps. In this context, there must be a language to express the desired information, including spatial relationships between objects, in order to specify the desired objects in a query to a GIS (e.g., show all the camping areas north of Bangor). In order for such a facility to work, the human understanding of the spatial relationships used, and the formal definitions applied by the system, must be very similar, otherwise the system will not retrieve and display the desired information. This may lead to frustration of the user if detected, or to erroneous judgement by the user if not detected.

6.3.2 Navigation Aids

The use of GIS to help humans navigate in complex situations seems to be an obvious application. Experiments with systems to navigate in towns to find locations specified by street address are underway, and the first commercial systems are available. It is expected that most cars in the future will include some navigational aid.

In such a system the user must be informed about spatial actions ("turn left at next red light"). It is not yet clear what type of concept for expressions of spatial relationships is most effective in this situation.

7. Possible Approaches

In this section we rapidly enumerate a number of approaches that are currently being tried without assessment.

7.1 Other Number Systems

The work currently done at the Renselear Polytechnic Institute by W. Randolph Franklin attempts to overcome the problem that real numbers used in finite computer systems cannot correctly represent geometric situations. Rounding of computed coordinate values cannot be avoided. They study other number systems, e.g., (big) rational numbers [Franklin and Wu 1987] or number systems including roots of polynomials, which could overcome these limitations [Franklin 1984].

7.2 Raster Systems

Operations in current raster systems are based more on opportunity than on mathematical analysis. A most extensive treatment is found in Tomlin [1983], but a more compact notation of the algorithm could show similarities in the proposed operations. Approaches to build an algebra of raster operations are shown [Serra 1982].

7.3 Combinatorial Topology

The application of combinatorial or algebraic topology to GIS has a long tradition [Corbett 1979]. Combinatorial topology offers concepts to deal with the topological relations between spatial objects in a symbolic form. By using a restricted form of a

simplicial complex, we found that qualitative aspects of most of the metric properties can be included [Frank and Kuhn 1986].

8. Conclusions

Understanding spatial situations and processes is a primary goal of most studies in geography. We clearly need a formally defined vocabulary that allows us to discuss and reason on these processes. By spatial theory we understand a set of concepts of abstract space and abstract operations on space, which are formally defined; however, such concepts are only valuable if they correspond to human beings' understanding of space. It is thus necessary to combine the formal theoretical studies with results from cognitive scientists, who analyse the 'natural' concepts we use.

Acknowledgement

This work was partially funded by a grant from NSF under No. IST-8609123. Thanks to Jeff Jackson and Max Egenhofer who helped prepare this paper.

References

Boyle, A.R. (1983). Final report of a conference on the review and synthesis of problems and directions for large scale geographic information system development. In: NASA Contract NAS2-11346, April.

Cebrian de Miguel, J.A. and J.B. Sendra (1987). Microordenadores en Geografia. In: 1 *conferencia Latino america sobre informatica en Geografia*, San José, Costa Rica, October.

Chrisman, N. (1975). Topological data structures for geographic information processing. In: *AUTO-CARTO II.*

Corbett, J.P. (1979). Topological principles of cartography. Technical Report 48, Bureau of the Census, Department of Commerce.

FIG (Fédération Internationale des Géomètres) (1981). In: *XVIe Congrès International des Géomètres*, Monteux (Switzerland).

Frank, A. (1987). Overlap processing in spatial information systems. In: *Eighth International Symposium on Computer-Assisted Cartography*, ed. N.R. Chrisman, Baltimore, Md.

Frank, A. and W. Kuhn (1986). A provable correct method for the storage of geometry. In: *Second International Symposium on Spatial Data Handling*, Seattle Wa.

Frank, A. et al. (1986). Formal methods for the accurate definition of some fundamental terms in physical geography. In: *Second International Symposium on Spatial Data Handling,* Seattle Wa.

Franklin, W.R. (1984). Cartographic errors symptomatic of underlying algebra problems. In: *International Symposium on Spatial Data Handling*, Zurich, Switzerland, August.

Franklin, W.R. and P.Y.F. Wu (1987). A polygon overlay system in Prolog. In: *Eighth International Symposium on Computer-Assisted Cartography*, ed. N.R. Chrisman, Baltimore (MD).

Hobbs, J. and R.C. Moore (1985). *Formal Theories of the Commonsense World.* Ablex.

Klein, F. (1872). Vergleichende Betrachtungen über neuere geometrische Forschungen (Comparing considerations about recent geometric researches), Erlangen.

Mark, D.M. et. al. (1987). Spatial terms and spatial concepts: Geographic, cognitive, and linguistic perspectives. In: *International Geographic Information Systems Symposium: The Research Agenda*, Crystal City, Va, November.

Moise, E. (1963). *Elementary Geometry from an Advanced Standpoint.* Addison-Wesley Publishing Company.

Peuquet, D.J. and Zhan Ci-Xiang (1987). An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition,* 20(1).

Robinson, V.B. and R.N. Wong (1987). Acquiring approximate representations of some spatial relations. In: *Eighth International Symposium on Computer-Assisted Cartography*, ed. N.R. Chrisman, Baltimore, Md.

Serra, J. (1982). *Image Analysis and Mathematical Morphology.* Academic Press Inc.

Tomlin, C.D. (1983). Digital cartographic modeling techniques in environmental planning. Ph.D. thesis, Yale University.

Walker, J. (1987). The amateur scientist: Reflections from a water surface display some curious properties. *Scientific American,* 256(1), January.

Zadeh, A. (1981). Test-Score semantics for natural languages and meaning representation via PRUF. In: *Empirical Semantics*, ed. B. Rieger, Brockmeyer, Bochum, pp. 281-349.

MANAGING STORAGE STRUCTURES FOR SPATIAL DATA IN DATABASES

Max Egenhofer
University of Maine
Department of Civil Engineering
103 Boardman Hall
Orono, ME 04469
U.S.A.
MAX@MECAN1.bitnet

ABSTRACT

Data driven, dynamical storage structures are well-suited for fast access on spatial data in databases. In general, spatial objects are stored such that neighbors are physically close to each other; however, these structures do not use information from a semantically richer data model and treat all spatial data similarly.

The structure managing access on spatial data reflects the distribution of the data. A collection of spatial data consists of heterogeneously distributed objects. Some of them are equally distributed on the area, some are concentrated at a few locations, others occur less frequently and less systematically. Most storage structures for spatial data are trees or similar structures; the deeper they grow, the less efficient is the search within large regions for widely-spread and less frequently occurring objects.

Data models which are part of databases provide details for different kinds of spatial data. This information about the structure of the data will be used adding some knowledge about the density of objects. This leads to a storage structure which is based on both the spatial location and extension of each individual object and the density of object classes. A specific structure, the Field Tree which is part of a database management system (PANDA), will be the centre of concentration.

1. Introduction

A large number of applications in modern information systems and geo-sciences deal with spatial data. CAD/CAM, geo-information systems, such as geographic information systems (GIS) and land information systems (LIS), and cartography are areas in which non-spatial data and spatial data merge. Due to the large amount of data in such systems, data must be stored in databases. Standard (commercial) databases provide sufficient support only for non-spatial data.

For several years it has been a field of interest to investigate storage structures for spatial data and there is a large variety of proposals for different storage structures, e.g. Guttman [1984], Nievergelt et al. [1984], Tamminen [1981], each of whom treated spatial objects slightly different. In Sikeler [1985] and Kleinert and Brassel [1986] a good overview of the different techniques is given. Those techniques are required to organize data such that an acceptable performance is guaranteed when selecting parts of them. There is an ongoing attempt to incorporate the treatment of spatial data into the whole process of database managing — from the low-level storage structures up to extending query languages by spatial aspects. Latest proposals for object-oriented databases incorporate specialized structures and access paths on complex objects into databases [Dittrich 1986, Orenstein 1986]. In the sequel, such a hybrid model, where non-spatial data and spatial data are incorporated in the same management system, will be discussed.

This paper describes some results of 5 years' experience with such a structure for two-dimensional spatial data, the Field Tree [Frank 1983], which is embedded in an object-oriented database management system (PANDA) [Frank 1982, Egenhofer and Frank 1987].

2. Spatial Objects

In the data model we use, data is structured such that similar objects sharing common operations form a class [Mylopoulos and Levesque 1984]. Objects are the occurrences (instantiations) of data describing something that has some individuality. In addition to non-spatial data, spatial objects are characterized by their spatial location and extension.

From the technical point of view, each object has a size which corresponds to the space it requires on some storage device.

2.1 A Storage Structure for Spatial Objects

Storage and access methods which are tailored to the needs of spatial data are required to efficiently answer user queries. It is likely that an object is several hundred times more often accessed than it is modified. This leads towards optimizing the access on spatial data.

Multi-purpose systems deal with a large variety of different spatial data. Besides points, structure for two-dimensional spatial objects must treat linear and areal objects as well. The emphasis lies on treating all of them equally, namely as spatial data, and not developing a structure for every different type.

2.2 Spatial Queries

Extraction of space related information out of a large data set most often needs data of a whole region. This is significantly different from queries on non-spatial data in which only specific objects are supposed to be retrieved fast. In general, range queries are difficult to predict and thus are not efficiently supported by databases. A typical query with spatial objects is

RETRIEVE ALL <objects> OF <region>

where the region may be either some arbitrarily delineated area or a spatial object representing an area, such as the 'State of Maine'.

When asking queries on spatial data, we assume that further information about the neighborhood is required, either immediately, or very soon. Immediately means within the process of the same query, right after treating a spatial object. For example, when drawing the boundary of a parcel, right after accessing the parcel, its delineating points and the shape of the lines in between them will be needed. Soon means that, having a

picture on the screen, additional information is desired. Such additional information, for example, might be completing next a map of all parcels by all houses.

These two concepts of expected data access must be incorporated into a suitable storage structure and their organization. The first one requires that several neighboring objects be accessed at once. The second concept requires that objects be provided which are accessed once for faster access times in forthcoming actions.

2.3 Filtering Spatial Data

The goal of a spatial storage structure is to provide fast response time on selective objects. Dealing with spatial objects and all the details of their shape is not beneficial. Spatial data can be totally irregular.

A more general and less detailed treatment of spatial objects allows us to simplify the selection at access time. Such a mechanism acts like a filter excluding all those parts which do not belong to the desired choice and offering a set which may be larger than the desired one. The goal of a filter is to quickly and easily reduce large sets to smaller ones which can be managed afterwards with more precise tools. The more accurate a filter represents the parts to be selected, the smaller will be the amount of irrelevant selections; however, a filter must never be too small, otherwise relevant parts might be excluded. Such filters can perform quite fast as they act on more general and simpler objects, hence, their operations are easier to perform. Hashing methods, for example, provide fast access on uniquely-keyed values by internally mapping objects on less meaningful item. Spatial data can be treated in a similar fashion.

Location and extension of a spatial object can be generalized to a geometrically simple, regular figure. The minimal spatial description for a two-dimensional object is a point and a circumscribing circle such that the whole object lies within the area delineated by the circle.

Other figures can describe the shape of an object more precisely with less overhead, but requiring more information. For describing a rectangle as the circumscribing figure of an object at least four values needed.

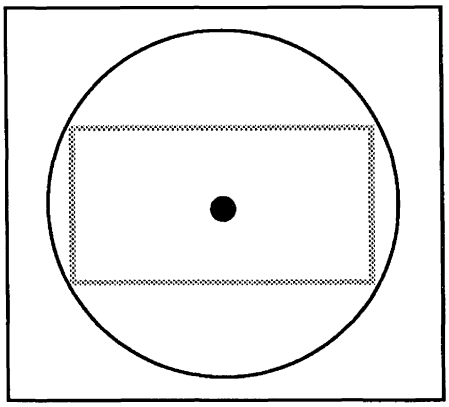


Figure 1. A point and a circle generalizing a spatial object.

2.4 Disk Access

Before presenting all these data as information, a large amount of data must be accessed from a storage device. The access must be fast and suitable for large data sets. Interactive communication requires answers within a few seconds, otherwise users get interrupted in their process of working and analyzing, loose concentration, or even get bored. Sequential search through very large sets is not acceptable for presenting only a small subset of the data.

For the following discussion, the assumption is made that large sets of spatial data are stored on a hard disk and that the whole amount of data is too large to be concurrently kept in main memory during processing. This assumption holds for even 'small,' densely populated regions where the amount of spatial data is fairly high.

A hard disk consists of a rotating cylinder and a read and write head. The units for reading data from a hard disk are so-called pages: buckets ranging from 256 bytes to 4 Kbytes. The access time for reading from a hard disk is composed of three parts:

- The time to move the read head to the appropriate track. This takes about 10 to 20 milliseconds.

- The wait time until the desired part of the track is under the head. On average, this takes half a rotation of the cylinder which is approximately 15 milliseconds.
- Finally, the time to read the data. The amount of data read is about 0.5 to 3 megabytes per second.

These numbers who that the actual time for reading even large data blocks is small compared to the time needed to place the head for reading. It is well known that the major influence on response time in database management systems is the disk access time and the number of physical disk accesses. This highly influences the strategy for reading spatial data from a hard disk. The goals are:

- Minimizing the number of disk accesses.
- Reading larger amounts of data at a time. This requires that spatial objects be organized such that these data read at a time are expected to be useful in the nearest future.
- Access to main memory is 10 ** 6 faster than access to a hard disk. Forthcoming access can be speeded up by keeping data which were read once from the hard disk in less expensive main memory. Such a treatment requires a buffer strategy reading data from the hard disk only if the data is not available in the buffered former accesses.
- Several pages which are frequently used should be buffered in main memory. This prevents the system from bad performance when data is distributed over several pages and not concentrated on a single page.

It is beneficial to not only read a single element per access, but to read several of them at a time. Therefore, spatial data must be structured and organized such that they are well distributed when read.

2.5 Towards a Suitable Structure

The concept of coherent neighborhood [Dutton 1978] requires the storage of neighboring data physically close to each other. Thus, accessing a spatial element implies that its neighboring objects are found with it. When keeping these objects locally (i.e., in main memory), faster access is provided for all forthcoming operations on the object itself or its neighbors.

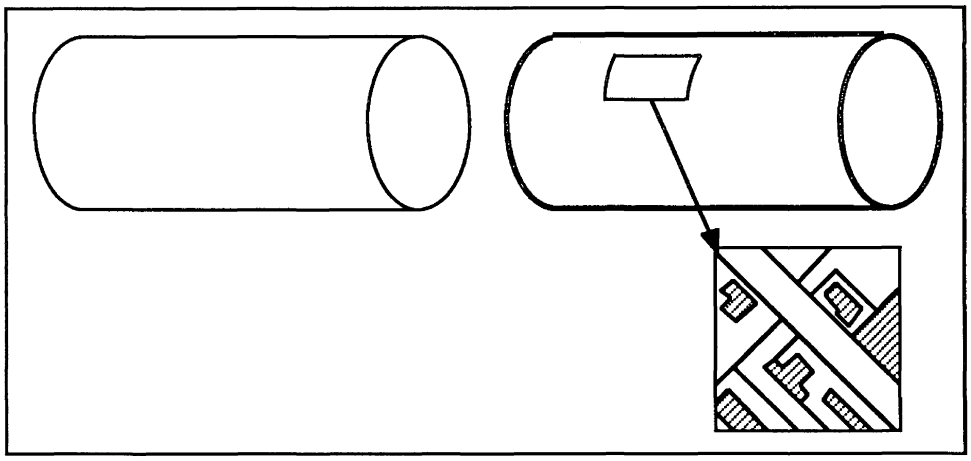


Figure 2. Storing neighboring objects close to each other.

Not all pages which are read in can be concurrently kept in main memory. A managing system deciding which pages to keep and which pages to cancel is an extrapolation from the events in the past to expected events in the future. A suitable method is the least-recently-used strategy (LRU). Data which have not been used for the longest time are expected not to be used in the near future, whereas data which have been recently used are supposed to be re-used soon. This concept fits well to the structure of expected queries: first, the user chooses an overview of a region, then he or she looks at a smaller part of the former representation in a more detailed view. Finally, one or several objects within this smaller part are processed by executing some interactions.

In conclusion, spatial data must be organized such that neighboring objects are close to each other. When reading a single object, a whole physical page will be read including the neighboring objects. This page is kept in a local buffer to which faster access is provided. Objects which are on a buffered page need not be read from the disk, but directly from the page buffer. Keeping several pages in the buffer allows the system to efficiently treat areas which are greater than the corresponding area of a single physical page.

3. Treating Generalized Spatial Objects

Spatial data is treated according to its spatial location and extension. The data structure managing spatial data should be independent from the type of spatial data and should treat all spatial objects the same way [Frank 1984]. Using minimal bounding rectangles (MBR) with sides parallel to the coordinate axes is an acknowledged way for a general treatment of spatial objects.

3.1 Minimal Bounding Rectangles

Rectangles are the Cartesian product of two perpendicular one-dimensional intervals. Their operations can be easily reduced to the combination of such intervals [Egenhofer 1987], providing a clean and clear algebra. Using MBRs instead of more detailed objects reduces the treatment of objects to a semantically lower level; MBRs have less meaning and less detailed structure than an arbitrarily shaped two-dimensional object.

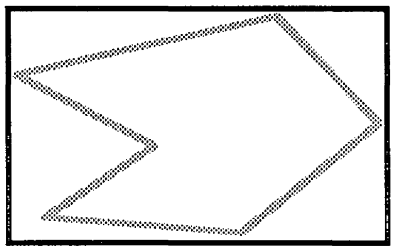


Figure 3. A minimal bounding rectangle (MBR).

The same applies for the search area: restricting the 'window' of a spatial query to a rectangle is a generalized way which can be easily performed. When uniformly treating the spatial location of objects by their MBRs, the access within a certain window is reduced to the simple operation checking whether two rectangles (the window and an MBR) have some common parts.

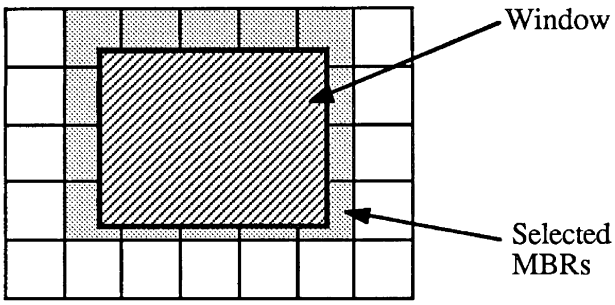


Figure 4. A query window and the affected MBRs.

MBRs are, at least for straight-line objects, easy to calculate as they are just the 4 extreme values of their coordinates.

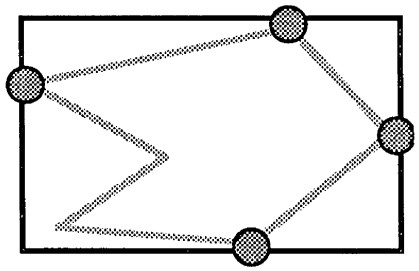


Figure 5. Constructing the MBR from the extreme values of an object.

The calculations get more complex for objects with non-straight boundaries such as arcs or splines; their extreme values might not be represented by the bounding points and, thus, have to be calculated expensively.

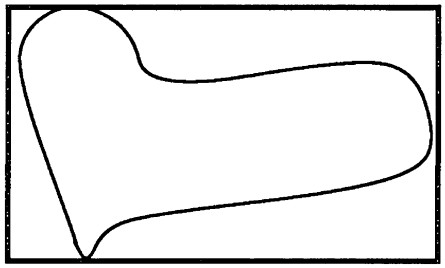


Figure 6. Calculating the MBR for an object with non-straight segments.

It is important that the whole object lies within the MBR and not even a very small part of an object must be outside of its MBR. Otherwise, searching fails in cases where the query window touches only those parts which lie outside of the MBR.

An alternative is to choose larger MBRs avoiding such computations. However, the larger the MBRs, the more space that is wasted with non-object parts and the more additional effort that is needed at selection time. The less difference there is between an object and its MBR, the better the performance at selection time.

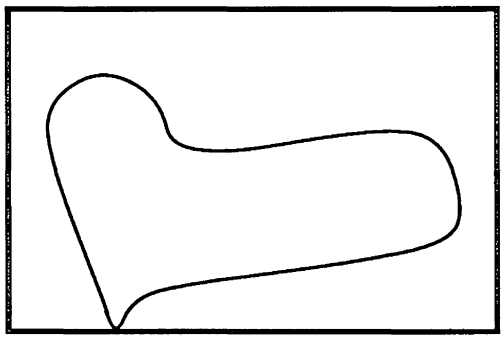


Figure 7. An alternative non-minimal bounding rectangle.

Using MBRs as the generalized spatial object allows us to treat points, lines, and areas equally, independent of their shape or type. Of course, MBRs do have their disadvantages. All objects returned from a search within a window do not necessarily fall within the area searched for. There might be objects which are outside, but their MBRs overlap with the search window. Since MBRs act as a filtering mechanism, the set of objects returned from a range search can be greater than the set of affected objects.

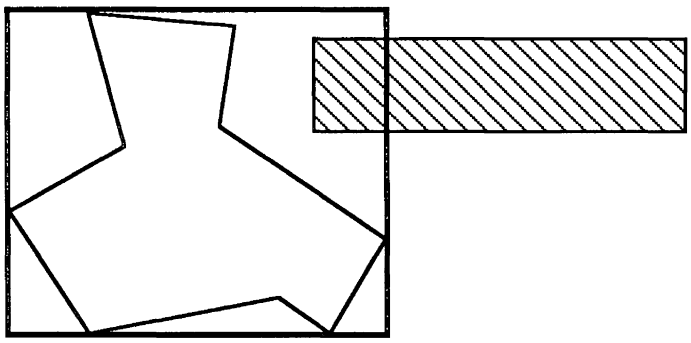


Figure 8. A window and an irrelevant object.

The size of an MBR depends on the orientation of the corresponding object. The MBR in Figure 9a is as big as the MBR of the area in Figure 9b. MBRs do not represent all properties of spatial objects, they only represent their maximum extension. This influences the range search. Although both objects in Figure 8 are the same, their MBRs are significantly different. The smaller the difference between the area an object covers and its MBR, the better it is approximated by the MBR, and the less it will be accessed in vain.

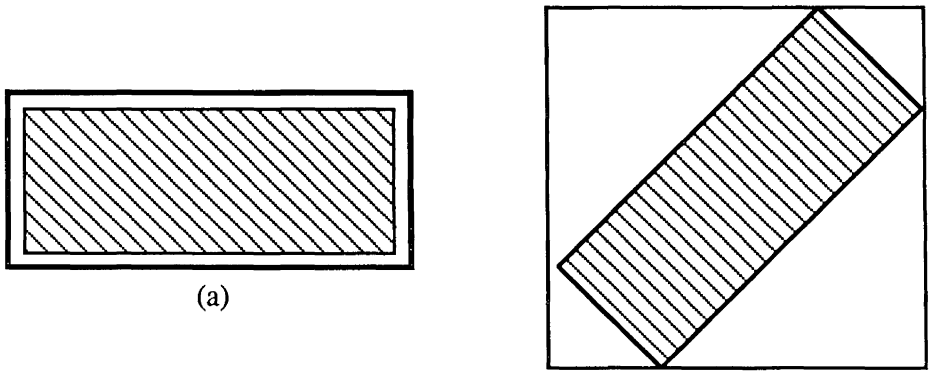


Figure 9. The MBR depends on the position of the object.

3.2 On Splitting Objects

It might be worthwhile to split the objects into smaller object-parts and, therefore, get smaller (and more precise) MBRs [Scheck and Waterfeld 1986]. Such a strategy requires knowledge about the specific object, namely, where and how to split it. This depends not only on the class of objects, but also on its very specific geometric shape. The abstract and pure mathematical treatment of MBRs cannot be applied any more. Allowing the objects to be split would move the treatment of spatial objects to a semantically higher level. There the treatment would not be restricted to MBRs, but would be extended on individually different objects.

For example, splitting the object in Figure 10 such that two MBRs cover a minimal size would definitely reduce the failure of undesired selection. On the other hand, it is

required that the interval operations be merged with some additional knowledge of the internal representation of each object, namely, how and where to split the object.

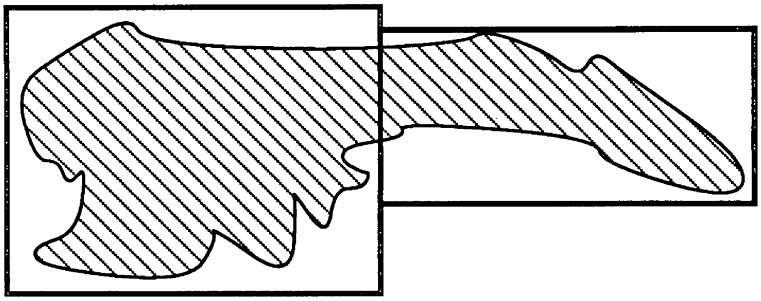


Figure 10. Splitting an object into two parts.

The splitting of spatial objects is time consuming as it requires some intersection operation. Moreover, at retrieval time objects which were once separated must be composed to form a single object.

The major problem with splitting is the co-existence of spatial and non-spatial object parts in databases. Splitting objects separates spatial from non-spatial information. When splitting an object, where do its non-spatial attributes go? Storing them at each object-part is surely not desired as it produces redundancy and overhead. Dividing the attributes equally over all parts is not reasonable either. What, for example, would be the house number of half a house? Finally, storing the non-spatial information at another location leads towards having at least two disk accesses per page: one for the page with the spatial data, and another one for the non-spatial attributes.

An example may show how often spatial and non-spatial data are needed in parallel. Drawing a map with some features such as symbols for points, different line types, areas filled with corresponding patterns, and labelling points or areas requires interaction between spatial and non-spatial data. When drawing a feature, the location is needed, drawing a line needs the feature code, and filling the area needs the individual attributes to choose the appropriate filling pattern.

We conclude that objects should be preserved and not be split into several smaller parts.

4. The Field Tree

The Field Tree [Frank 1983] is a storage structure suitable for two-dimensional spatial data. It was developed for treating geographical objects in land information systems (LIS) and their specific properties. Geographical objects are abstract spatial objects which are composed of smaller spatial objects of different types (e.g., area, lines, points). Geographical objects can be distributed in varying density, and data can have both spatial and non-spatial properties.

There are two specific differences between the Field Tree and most other spatial storage structures. First, it treats spatial data as a 'property' of data allowing the user to concurrently treat spatial and non-spatial data in the same management system. The benefits from this kind of treatment are that spatial and non-spatial data are not separated from each other.

The other major difference is the way the logical structure for spatial data is built up. Similar to other tree-like structures for spatial access such as quadtrees [Samet 1984], the Field Tree organizes the space in smaller parts (subfields) which are generated when a field overflows; however, in the Field Tree the next lower level is shifted by half the extension of a subfield both to the right and to the top.

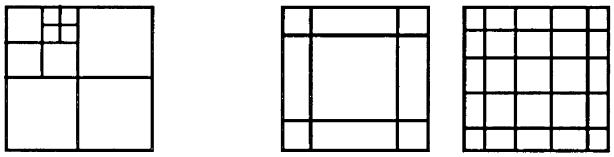


Figure 11. Subdividing in a quadtree (a) and the Field Tree (b).

The areas of two different levels overlap such that objects crossing the boundaries of a field do not cross the boundaries in the next lower level. Each field has one major

subfield, four subfields which it shares with either of its four neighbors, and four subfields it shares with two direct neighbors and one neighbor in the diagonal.

The subdivision of a field is not inherently aligned to the former field, but shifted. Thus, objects crossing the boundaries of some subfields are not prevented from being moved to a lower level. The boundaries of a field do not coincide with the boundaries of any of its subfields. Their subfields, the sub-subfields, will have boundaries which do not coincide with all of their former boundaries either. This implies that for a small object crossing the boundary of its next subfield, another subfield will exist to fit in. Larger objects which do not fit on a subfield or sub-subfield cannot be moved onto a lower-level field.

The Field Tree organizes spatial data according to the structure of the database (schema). The relevant part for the geometry of an object is its MBR, and in addition each MBR is related to the corresponding object class. Spatial access on one type or several types of objects can be performed faster by excluding all those rectangles which are not related to the object class for which a search is currently underway.

Objects are spatially accessed by searching in the field tree for the leaves which overlap with the search window. This process is performed in a top-down manner. In order to find all spatial objects of a specific type, all overlapping fields must be searched through.

5. Data Distribution

In this section we will concentrate on the distribution of spatial data and the consequences for a suitable structure.

5.1 Heterogeneously Distributed Data

Spatial data are not homogeneously distributed. They rather concentrate at some locations and are absent in others. A glance at a map of the State of Maine with all major towns shows a concentration on the coast and a very low density in the northern parts.

Figure 12. The distribution of towns in the State of Maine.

A spatial storage structure must perform well under such circumstances. Dynamical data structures keep track of both homogeneously and heterogeneously distributed data as they can be extended where data are concentrated and they are minimal where no data occur.

5.2 Data Density

The density of spatial data depends on two issues:

1. Topography increases the amount of data if abrupt changes in the natural environment occur providing a higher amount of perceivable differences than continuous changes.
2. Concentration of population increases the amount of data as humans change nature significantly. The more people that are concentrated on some locations, the more they influence nature and the more new spatial features, such as houses, roads, etc., are built.

Different data types have a different frequency, for example, within a town, a great number of residences and business houses occur, while the number of hospitals, railway stations, etc., is relatively small. When all these buildings are treated the same way, access on hospitals alone is not very effective. As they are merged with all other

buildings, a great number of fields have to be searched through in order to find the few hospitals.

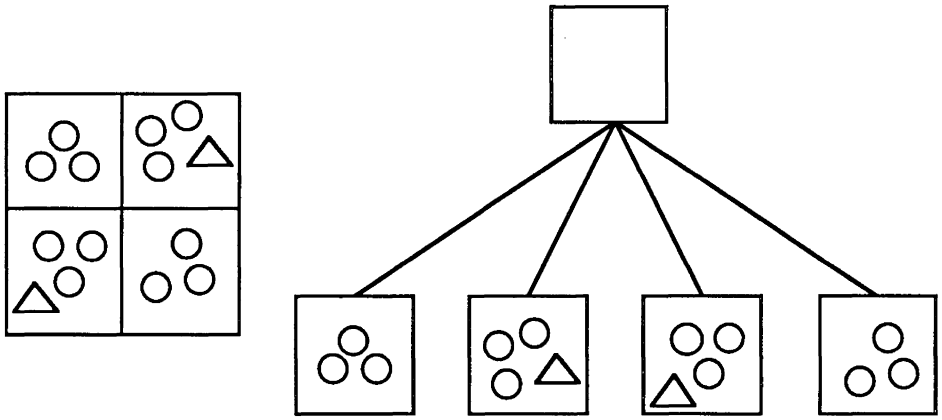


Figure 13. Two object classes of different density.

It would be rather useful to store the rarely occurring objects on a higher level than the objects which are densely distributed. With such a knowledge, the Field Tree can be more effectively organized and is expected to provide faster access on specific object types within large areas.

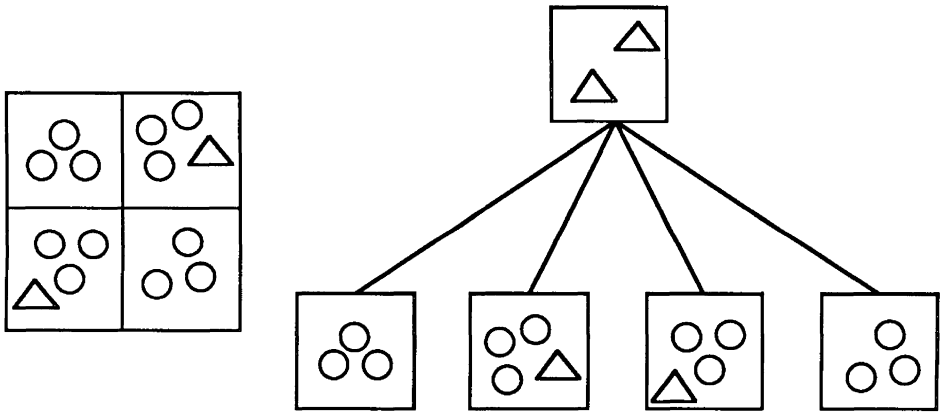


Figure 14. Separating object classes with different frequencies.

When we attach a 'density' attribute to each object type (class), we can optimize the structure for each class such that useless disk accesses can be avoided. The density attribute is represented by the value of the lowest possible level to which an object may

be moved in the Field Tree. The more compact and closer the data are expected to be,. the greater the value for the density attribute. In the Field Tree, a lower density value prevents all objects from being shifted too many levels down. The search for objects of a certain class can be cut off when reaching its minimum level. This reduces the number of accesses on objects of this type by the number of all lower fields.

We predict the density for object classes from observations and expectation of how data will be distributed. An area worthwhile to investigate is how to evaluate density and distribution of spatial data.

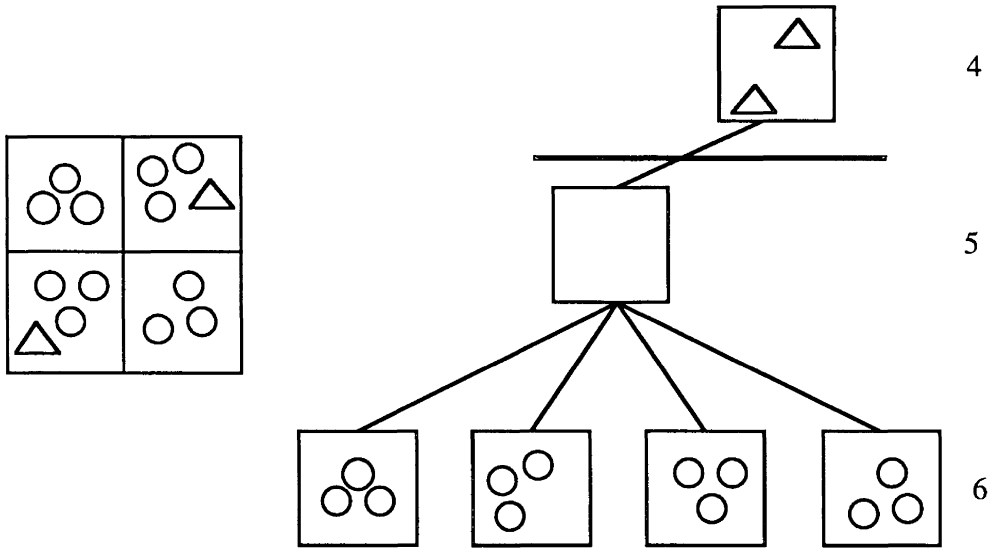


Figure 15. Cutting of a query on objects with a density level 4.

6. Strategies

Finally, we analyze the subdividing of fields. This is the part of the Field Tree discussion which is most transparent to the user when dealing with large data sets. In this section, we will address some problems which are related to embedding a spatial storage structure into a database.

6.1 Balancing the Size and the Number of Fields

We could imagine two extreme situations, one with all objects on a single field, another with a separate field for each object. It is obvious that both cases are not desired. The strategy instead looks like having fields which are equally populated with a number of objects, and which can be quickly searched through. Fields which are only slightly populated will decrease performance due to the numerous accesses, while fields which are highly populated will decrease performance by long sequential searches.

6.2 Subdividing Fields

A field corresponds to a physical page which is a limited amount of storage space. This restricts the number of objects which can be stored on a single field. Once a field is filled with the maximum number of objects, it has to be subdivided into subfields creating a new physical page for each subfield and spreading the corresponding objects on the new subfields. Objects which are too big to be placed on a subfield have to stay on the original field. The same holds for objects which cross the boundary of a subfield unless a sub-subfield exists which covers the overlapping region between two neighboring subfields.

A strategy for subdividing fields looks as follows: Only those subfields which are designated to contain some objects are subject to being created. We will not create empty subfields as they would increase the overhead without adding any benefits.

Considering the balance between the size and the number of objects per field, we apply two rules when subdividing fields:

1. Create only the subfield which will contain the most objects and move them down. Leave all the other objects on their current level.

2. Create such a subfield only if at least a certain number of objects can be shifted down. This prevents the creation of sparsely populated subfields. If a field cannot be reasonably subdivided, instead create a parallel page. A parallel. page is another physical unit which is assigned to a field in addition to its corresponding page.

6.3 Starting Data Collections

When starting to load data into the structure, only a single field, the root, is available. This field must be subdivided as soon as it overflows. When storing data, the sequence in which they are loaded does not always lead to an optimal subdivision. When starting from scratch, we observed that the first data sets are concentrated at a few locations. For concentrated data, subdivisions in the way described above are not very beneficial as they just shift all of the data one level down requiring another subdivision. This continues until a subfield size is reached which is smaller than the area all objects cover. Therefore, we propose to start with a pre-built part of the Field Tree with half the level of the maximum density value. Once a major data set is stored, the tree will be reorganized such that unused subfields are cancelled.

6.4 Self-Organizing vs. Reorganization on Demand

At first glance, dynamical data structures seem to be the remedy for the problems arising from heterogeneously distributed data. They optimize themselves whenever the structure is changed. This means that the structure can be internally updated without any influence of a user. Such updates, however, might be visible to the user whenever updates occur during some 'simple' modifications which are supposed to be done quickly.

The alternative to self-organizing structures is reorganization on demand. In such a system, the structure is only changed when the user is aware of a time consuming operation. Such considerations are crucial for designing human interfaces. A system reacting differently and spending time every now and then on operations which are not transparent to the user is confusing and not very trustworthy.

7. Conclusion

In this paper, we concentrated on some deficiencies of the organization and management of structures dedicated to support storing spatial data. We addressed preserving spatial

objects and not splitting them in some subparts, not to separate spatial data from non-spatial data, and to consider heterogeneously distributed spatial objects.

8. Acknowledgement

Thanks to Jeff Jackson who helped prepare this manuscript.

References

Appelrath, H.-J. (1985). A geo-concept of an application-neutral geographic database system and its implementation as an INGRES-frontend (in German). In: *Database Systems in Office, Technique, and Sciences*, Karlsruhe, F.R. Germany.

Dittrich, K.(1986). Object-oriented systems: The notation and the issues. International Workshop in Object-Oriented Database Systems, Pacific Grove, California.

Dutton, G. (Ed.)(1978). *First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems.* Harvard Papers on Geographic Information Systems, Harvard University, Cambridge, Ma.

Egenhofer, M. and A. Frank, (1987). PANDA: An object-oriented database based on user-defined abstract data types. Report No. 62, Surveying Engineering Program, University of Maine.

Egenhofer, M. (1987). An algebra on linear intervals. University of Maine, Orono (forthcoming).

Frank, A. (1982). PANDA, A Pascal network database system. In: *Proceedings of the Fifth Symposium on Small Systems*, ed. G.W. Gorsline, Colorado Springs, Co.

Frank, A., (1983). Problems of realizing LIS: Storage methods for space related data: The Field Tree. No. 71, Swiss Federal Institute of Technology, Zurich, Switzerland.

Frank, A. (1984). Requirements for database systems suitable to manage large spatial databases. In: *Proceedings of the International Symposium on Spatial Data Handling*, ed. D. Marble et al., Zurich, Switzerland.

Guttman, A. (1984). New features for a relational database system to support computer aided design. Memorandum No. UCB/ERL M84/52, Electronic Research Laboratory, College of Engineering, University of California, Berkeley.

Kleinert, A. and K.E. Brassel (1986). Hierarchical grid structures for static geographic data bases. *Proceedings, Auto Carto*, London.

Mylopoulos J. and H.J. Levesque (1984). An overview of knowledge representation. In: *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, eds. M.L. Brodie et al., Springer Verlag, New York.

Nievergelt, J., et al. (1984). The GRID FILE: An adaptable, symmetric multi-key file structure. *ACM TODS 9.*

Orenstein, J. (1986). Spatial query processing in an object-oriented database system. *SIGMOD Record*, Vol. 15, No. 2.

Samet, H., (1984). The quadtree and related hierarchical structures. *ACM Comp. Surv.*, Vol. 16, No. 2.

Scheck, H.-J. and W. Waterfeld (1986). A database kernel system for geoscientific applications. *Proc. Second International Symposium on Spatial Data Handling*, Seattle.

Sikeler, A. (1985). , Examination of storage structures for 3-dimensional objects (in German). University Kaiserslautern.

Tamminen, M. (1981). Management of spatially referenced data. Report HTKK-TKO-A23, Helsinki University of Technology, Laboratory of Information Processing Science, Helsinki.

Discussion

Question: How do you define 'acceptable performance'?

Answer: It should be fast enough for interactive query processing.

Question: Using the cache memory will only speed up accessing during the second time around?

Answer: The initial access time is expensive, while the second time through will be much cheaper.

Remark: For the bounding area of a feature, the bounding rectangle is chosen instead of a circle as it is easier to calculate its intersection and also, they match each other better at the edges.

Remark: A point can be depicted as a rectangular box or an error representation (e.g., an error ellipse) surrounding it.

Question: What is the criterion used to split the data into levels, i.e., density of data or data type?

Answer: The data is organized by both the data density and the data type. For example, the hospital is of level 2, whereas the residence is of level 8 because there are fewer hospitals than residences in a given area. In the internal data organization, the physical page is being split up to accommodate for more data when it is filled. The data are then being reshuffled to correspond to the new fields. When accessing the hospital information, we only have to search from the root up to level 2.

Question: Is there any data on the nodes apart from the leaves, i.e., the hospital is on the node instead of a leaf, and for each page there will be some real data and pointers leading to the subsequent levels?

Answer: Yes, a single, logical field may be comprised of several physical pages of data. The amount of leaves and nodes is relatively large. As a result, a hospital which neighbours a residence may not be stored in the same disk page but it is definitely on the same path down towards the leaves.

Question: What is the irregular subdivision process, i.e., field tree?

Answer: This subdivision process is similar to quadtrees except that each sector is shifted by a half section eastward and southward.

Remark: Unlike the quadtree, this method avoids the splitting of objects:

1. it avoids splitting of an object as that is expensive;
2. in order to split, we need some characteristic information of the object prior to the splitting process.

Question: Does the line duplicate itself in the subdivision process?

Answer: No, there are no two lines that overlap one another. The grid used is a regular grid.

Question: What is the other advantage of field trees?

Answer: Its main advantage is that there is at least one field into which the object's geometry fits completely and thus remains undivided.

Question: Are the spatial and non-spatial data stored in the same place, and how do you store a line which is a boundary of a street and a parcel of land?

Answer: The line is considered as an object per se. The street and the parcel are objects, too, but they are stored at a higher level than the boundary object.

Question: Do you need a different shift for each object?

Answer: No, each object has a minimum bounding rectangle (MBR) and each MBR has the density attribute of its object class which corresponds to the maximum level of storage for the object.

Remark: The MBR of the objects may overlap each other.

Remark: Another spatial storage structure proposed by Guttman is the R-tree which has a similar MBR concept . However, the data are stored into records rather than layers. The tree is balanced by using the existing MBR values.

Question: Do you need some sorting algorithms for the input stage?

Answer: Yes, an algorithm is used to sort out the incoming data into their respective levels.

Remark: The search and delete processes are the most time consuming.

Remark: In a GIS application, we are dealing with a lot of spatial data together with a tremendous amount of attribute data. The objective of this exercise is to speed up access time and, probably, empirical studies may show us that it is worthwhile to segregate out the unimportant non-spatial data into another disk space.

Remark: In order to improve database performance, we should break up the highly used data into smaller fragments and a large appendix for the seldom used data. The deciding factor is to find the optimum criterion for segregation.

Remark: Spatial statistical theory is not being developed to study spatial queries. It is primarily used to study crop pattern, crop interpretation, etc. Spatial query generally exhibits a non-random statistical distribution.

Remark: Indexed data structure alone is insufficient for interactive mapping needs.

Remark: The field tree is incorporated as part of the database to speed up access.

Question: How about using the most frequently used information strategy instead of the least frequently used method?

Answer: There is no significant difference in access time independent of whichever buffer replacement strategy is used.

Question: Is it possible to use a time domain instead of a spatial domain basis for keeping the data in the buffer?

Answer: This can be done, but it is not worth the effort as the spatial domain approach can be improved by using more buffers.

Remark: This package utilizes the topology information to retrieve information on object basis. It draws information from the source of interest and then propagates toward the neighbouring information.

TOWARDS AND EXTENDED SQL FOR TREATING SPATIAL OBJECTS

Max Egenhofer
University of Maine
Department of Civil Engineering
103 Boardman Hall
Orono, ME 04469
U.S.A.
MAX@MECAN1.bitnet

Abstract

Current conventional query languages for databases such as SQL or Query-By-Example deal exclusively with alpha-numeric type data and do not provide support for specific queries on spatial data. There is a considerable trend towards combining non-spatial data with space-related data or even founding information systems such as GIS/LIS and CAD/CAM on geometry. Besides a support from the databases, this incorporation of spatial data requires that the user's interface must provide suitable tools for interacting with spatial data.

In this paper, we will propose in detail how SQL, a conventional query language, should be extended in order to be able to deal with spatial data. First, what the components of a query language are and what tools a user should have to query databases is analysed. The structure of SQL as relational calculus is investigated and compared with the structure of an object-oriented model, a more advanced and sophisticated data model than the relational model. Subsequently, the specific properties of spatial data are presented, and it is proposed how to incorporate them into the current SQL syntax without adding major changes.

1. Analysis of User Queries

In this chapter we will analyse what a query language should treat and what its parts are. We will investigate what the user should be able to address from a query language and

what information the database needs in order to access data. We will look at data retrieval and presentation in a more general and more advanced matter than the relational model does. The object-oriented approach concentrates on objects and relations between objects without any dealing with more details of the objects. This is different from the way the relational data model structures data. In the following, we will see that the object-oriented approach allows the user to deal with data with less or even without any knowledge of the internal structure of the objects.

1.1 A Query To A Database

In an object-oriented view, a query to a database consists of

- addressing the object types with which the query deals,
- specifying the constraints on the objects (this includes constraints between objects),
- specifying the fields for extract by the query,
- specifying the type of the output display and the display device,
- specifying the output format.

These steps somehow reflect the way humans extract information from traditional filing systems. The query corresponds to a request to someone else to search the existing files for some data. The person doing the search first has to choose the files where the related data is archived, then he or she picks the referenced data and combines them with others using cross-references or parts of the data found. Finally, he or she takes the requested information from the selected data and brings them in a form such that they are most useful for the purpose of the requestor.

2. Analysis of SQL Statements

SQL is a structured query language for relational databases [Chamberlin et al., 1976] and was introduced as the query language for System R [Astrahan et al., 1976]. Due to the underlying relational data model [Codd, 1970], SQL deals only with relations and

combinations of relations, and it can be even seen as a purely relational algebra and a relational calculus with 'syntactic sugar' added [Korth and Silberschatz, 1986]. Any query in SQL results in a relation (including the empty relation). For an intensive discussion on the relational data model, see Ullman [1980].

SQL is not only a 'query' language (where query is understood in its proper sense), but also a manipulation language allowing the user to both inquire about the current state of a database and to change this state by adding, deleting, or modifying its contents. In this section, we will concentrate on the basic 'query' aspects. The needs for an appropriate data manipulation language will be discussed in another paper.

An SQL-query can be separated into 3 parts:

- The SELECT clause corresponds to the projection operation of the relational algebra and is used to specify the attributes to be included in the output list from the items chosen.
- The FROM clause designates a list of the relations with which the query is concerned.
- The WHERE clause imposes conditions between a relation and a value or between two relations.

2.1 SELECT Clause

The SELECT clause specifies how the resulting table will look. On the one hand, SELECT is a filter passing only the values of the specified attributes to the next higher level; on the other hand it determines the sequence of the attributes in the table.

SELECT has two operational modes dependent on the level of the query. A 'SELECT x, y, z' on the main level means 'make a table with the attributes x, y, z' and implicitly write the table to the output device. The same statement on a lower level, however, would only influence the composition of the table to be passed without any direct influence on the output.

2.2 FROM Clause

The FROM clause specifies the tables (relation names) which must be retrieved for combining and filtering. The relation names in a FROM clause correspond to the former SELECT clause and the following WHERE clause(s).

In general, all attributes stated in a SELECT must be part of the relations mentioned in the following FROM. The same applies to the WHERE clause in which only attributes which were mentioned in the former FROM clause may be used.

Regulations which are defined in subqueries are local, i.e., they are only 'visible' in the subquery itself and in its directly following subqueries. Thus, 'local' relations can be 'global' for all following, nested subqueries.

2.3 WHERE Clause

The WHERE clause imposes conditions on relations. A condition is a combination of two predicates with an operator where the predicates are either a specified value, the value of a field in a relation, or a value evaluated by some function.

Several conditions can be combined with the logical operators AND and OR. NOT allows the negation of a condition or of a combination of conditions.

The operators for the conditions include the relational operators =, >, >=, <, <=, <>. Inclusion in a set can be checked by IN, ranges are stated by BETWEEN .. AND ... . LIKE is a weaker EQUAL comparing similar strings and substrings.

Note that the predicates in the WHERE clauses always act on attributes, not on relations. This requires that the user has detailed knowledge about the structure of the relation(s) he deals with. So, in what he is doing is comparing relations by comparing their detailed parts. For example, the query on all persons smaller than 6'3" must be expressed by a predicate on the height of persons, namely

WHERE height < 6'3"

and cannot be expressed by

WHERE person smaller than 6'3" .

These restrictions will be essential when defining operators for spatial objects.

2.4 Subqueries

Subqueries are parts of queries which form a complete query by themselves; however, the result of subqueries is not processed to some output, but contributes to a condition in a higher-level query part. SELECT can be used for subqueries contributing to the conditions in WHERE clauses. The very first SELECT- statement of a query is overloaded as it implies that both a relation is returned as a result and the result is printed on the screen. Formatting parameters can be added to the SELECT clauses allowing the user to structure the output. They include:

- DISTINCT for elimination of duplicates.
- ORDER BY allows the user to define the output to appear in sorted order.
- GROUP BY allows the user to compute functions of groups of tuples. The functions to compute include AVERAGE, MINIMUM, MAXIMUM, TOTAL, and COUNT. It is possible to state conditions that apply to groups rather than to relations by HAVING clauses.

2.5 SQL — A Relational Query Language

How close this SELECT - FROM - WHERE pattern is to the operations on a relation can be easily seen by looking at a table and how data can be extracted from it. A table is a representation of a relation with the colons being the structure of each item (occurrence) and each row representing one item.

HOUSE:	Number	Type	Color
	15 30 156	residence residence office	white yellow gray

Figure 1: Table HOUSE with the colons number, type, and color.

The FROM clause determines the name of the table to choose from. The WHERE clause specifies the conditions on the rows, i.e., it determines which items (rows) to select. Finally, the SELECT clause determines which colons to take from the result. For example:

```
SELECT number, type
FROM house
WHERE color = 'yellow'
```

determines to take the table HOUSE (FROM house), therefrom all the rows satisfying the condition (WHERE color = 'yellow').

number	type	color
30	residence	yellow

Figure 2: The relation HOUSE with all items satisfying the condition color = 'yellow'.

The resulting relation must be reduced to a table containing only the colons number and type (SELECT number, type).

number	type
30	residence

Figure 3: The relation HOUSE with all items satisfying the condition color = 'yellow' and the desired fields extracted.

The 5 functional operations of relational algebra (Cartesian produce, projection, selection, set union, and difference) are incorporated in SQL as follows [Korth and Silberschatz, 1986].

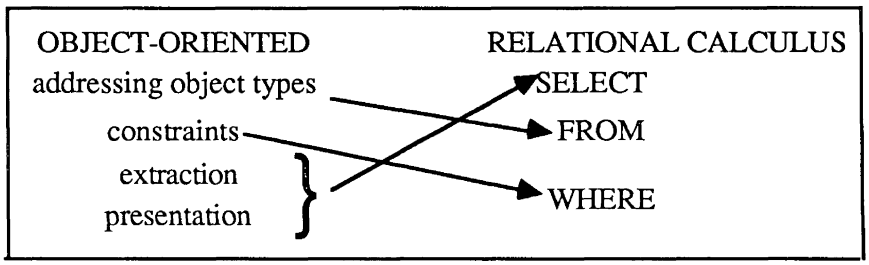
- The Cartesian product is represented in the FROM clause.
- Projection is performed in the SELECT clause.

- Algebraic selection is represented by the WHERE clause.
- Set union and set difference appear explicit in the syntax of SQL.

2.6 SQL Vs. Object-Oriented

Comparing the SQL-syntax with the object-oriented query design, we see two major differences:

1. The sequence in processing a query is somehow inverted starting with the last two issues (extracting and presentation), then addressing the objects, and finally formulating the conditions.



2. The SELECT clause is used for three different purposes: the extraction of the desired object parts, the destination of the output, and the specification of the representation.

   This overloading is an obstacle when extending SQL for two major issues: (1) treating objects which can have representations other than just alphanumerical printings, and (2) using the results for different purposes than creating a completely new output.

3. Spatial Data

In an object-oriented data model, data is structured such that similar objects sharing common operations form a class. Objects are the occurrences (instantiations) of data describing something that has some individuality. In addition to non-spatial data, spatial objects are characterized by their spatial location and extension.

From the mathematical theory of simplicial complexes we learned that each spatial object has a dimension. This dimension corresponds to the space in which this object is the elementary building block.

The presentation of spatial data can be different from the presentation of non-spatial data as spatial data can be graphically presented reflecting their spatial location and extension. Besides some statistical graphics, non-spatial data is presented exclusively as alphanumerical output. Moreover, combinations of graphical and alphanumerical output are needed in order to satisfy queries which require labeling of objects in a graphical presentation.

3.1 Properties of Spatial Data

Properties of objects are described as relations either between an object and a value or between objects. Such binary relations are of the Boolean type either being true or false. Very common are relations such as equal or greater than. Spatial data own additional properties which are specific for them and cannot be shared with non-spatial data. These relations are independent of the dimension of the object and thus can be applied for any space. We distinguish two types of relationships: topological relations and metrical relations. Topological relations are invariant under continuous transformation, while metrical relations depend on the metrical space. In the metrical space, we distinguish two relations between objects which are disjoint from each other: distance and direction.

In order to suitably combine and manipulate spatial data from an SQL interface, these spatial properties must be addressable by the user.

4. Towards Extending the SQK Syntax

In its current form, SQL is not well-suited for dealing with spatial data. The major deficiencies are:

- The specific relations of spatial data are not supported.

- The output facilities are supposed to be purely alphanumerical.
- No interactive interaction between a result presented and a query is incorporated.
- The results of a query always create a new output. No manipulation with the current output is possible.

It is obvious that these issues can only be incorporated by extending the syntax of SQL. The goal of changes in the syntax, however, is to add as few as possible new keywords, and not to change the overall structure of SQL.

4.1 Proposals

There have been several proposals on how to extend SQL such that it includes the treatment of spatial data; however, most of them lacked a consistent concept and were incomplete. We will give a short summary of the most significant proposals.

In Sikeler [1985] it is proposed to extend the syntax of SQL for two issues: (1) the treatment of spatial relations is included and (2) a picture list managing the graphical output is added. The spatial relations are strictly separated from the relations between non-spatial data by adding another clause ('WITH LOCATION') in which additional relations such as NEXT_TO are treated. The picture list is added to the 'attribute list' of the SELECT statement. The picture list is supposed to be the parallel instruction to the projection of the non-spatial data parts. In order to distinguish which parts to print and which parts to draw, the keyword GRAPHIC is introduced. Results of subqueries are passed to the next higher level to be fed into spatial relations by adding the postfix .LOC to the spatial objects indicating that the geometry (location) of the relation should be passed.

A typical query in this extended form would look like:

```
SELECT e.name, GRAPHIC (p)
FROM e, p
WITH LOCATION p NEXT_TO
        SELECT s.LOC
        FROM s
WHERE s.name = xyz
WHERE e.nr = p.name
```

Another extension of SQL is tailored to raster image processing [Roussopulos and Leifker, 1985]. PSQL (Pictorial SQL) combines direct spatial and indirect alphanumeric search. Each spatial object is extended by an attribute 'loc' for its location. This attribute is referenced in the SELECT clause for graphical output and in a specific clause for treating spatial relations.

The syntax is extended by two clauses; AT specifying the area to treat, and ON specifying a predefined output format (picture list). The picture list allows juxtaposition (synthesis) of dissimilar information of the same area from different sources.

The additional operations on raster cells are COVER, COVERED_BY, and OVERLAP. Additional functions are offered for calculating the area of an areal object and extreme points of lines or areas, such as NORTHERN and NORTHEASTERN.

A typical query in PSQL is:

```
SELECT city, LOC
FROM cities
ON us-map
AT LOC COVERED_BY (4 +/- 2, 6 +/- 3)
WHERE population > 45000
```

4.2 Extended SQL

In the following section, we will address extensions to the SQL syntax which are essential and obvious. We will show where they should be added and how the new syntax will look.

In order to provide solutions for the deficiencies we addressed earlier, we propose the following solutions.

4.2.1 Extending the FROM Clauses

The specific properties of spatial data must be supported by incorporating the topological and metrical relations into the SQL syntax. The spatial relationships correspond to the relationships between non-spatial data. Thus, the FROM clause is the

adequate place for adding the spatial relations. We propose to add the following, minimal set of relations between spatial objects :

1. topological relations

    1. disjoint
    2. contains
    3. inside
    4. meets
    5. overlap
    6. intersect
    7. common_boundaries
    8. equal

2. metrical relations

    1. direction: the relations for cyclic intervals (which correspond to the topological relations between spatial objects)

    2. distance: the conventional relations such as equal, less, greater.

We do not include 'fuzzy' relations, i.e., relations which cannot be strictly defined. Fuzzy relations are expressions such as CLOSE or FAR. First attempts on defining fuzzy relations between spatial data are promising [Robinson et al., 1985].

4.2.2 The Parameters of the Spatial Relations

Topological relations are exclusively between two spatial objects and there are no constant values which can be used as parameters in these relations. For example, it is not possible to formulate

WHERE parcel intersects 5.

At first glance, it seems feasible to define the spatial relations on objects directly. For example, the INTERSECT relation between two parcels could be expressed by the WHERE clause

WHERE parcel_1 INTERSECT parcel_2.

This approach is different from the conventional, relational formulations of conditions in SQL. In general, conditions are exclusively expressed on predicates. In order to achieve consistency with the relational model, we either have to express the spatial relationship on a field of a relation (attribute of an object) or on a value which is

evaluated by a function. For the first variant, a field (attribute) 'geometry' must be added to each space-related relation. The INTERSECT relation then looks like

WHERE parcel_1.GEOMETRY INTERSECT parcel_2.GEOMETRY.

This implies that GEOMETRY is a reserved word and must not be used for any other purpose explicitly. Using fields somehow implies that the values for establishing the spatial relation are directly and explicitly stored; however, such an assumption cannot be made. The geometry for an area, for example, is derived from its bounding edges.

The alternative is to express the spatial conditions by using a function evaluating the geometry of an object. The function GEOMETRY determines the geometry of an object such that it can be processed by the spatial relations.

The INTERSECT_relation then looks like

WHERE GEOMETRY (parcel_1) INTERSECT GEOMETRY (parcel_2).

5. Conclusion

In this paper, we investigated how to extend SQL in order to include the specific treatment for spatial data. We analysed the components of a query language in an object-oriented view and compared them with the structure of SQL, a purely relational calculus. We found out that SQL somehow inverts the single steps of a query by first addressing what to extract from the objects and then to state which objects to take and how to combine them. Moreover, SQL overloads the SELECT clause by using it for both the extraction of data and the direction of the alphanumerical output. As a result of this, any output is supposed to be printed on the current output device.

Spatial data own properties which are different from non-spatial data. Topological and metrical relationships between spatial data have to be added to the conventional conditions between non-spatial data. We proposed to extend the FROM clause in the SQL syntax by adding keywords for the spatial relations. Details of the extension showed again the difference between an object-oriented approach, where operations on objects can be formulated, and a relational calculus requiring operations on fields of relations.

Acknowledgement

This project was funded by a grant from Digital Equipment Corporation.

References

Astrahan, M.M. et al. (1976). System R: Relational approach to database management. *ACM TODS*, Vol. 1, No. 1, p. 97.

Chamberlin, D.D., et al. (1976). SEQUEL 2: A unified approach to data definition, manipulation, and control. *IBM Journal of Research and Development*, Vol. 20, No. 6.

Codd, E.F. (1970). A relational model of data for large shared data banks. *Comm. ACM*, Vol. 13, No. 6.

Korth, H. and A. Silberschatz (1986). *Database System Concepts*. McGraw-Hill Book Company.

Robinson, V., D. Thongs, and M. Blaze (1985). Machine acquisition and representation of natural language concepts for geographic information retrieval. *Proc. Pittsburgh Conference on Modeling and Simulation*, Pittsburgh, Penn.

Roussopulos, N. and D. Leifker (1985). Direct spatial search on pictorial databases using R-trees. Proceedings of International Conference on Management of Data, *SIGMOD Record*, Vol. 14, No. 4.

Sikeler, A. (1985). Examination of storage structures for 3-dimensional objects (in German). University Kaiserslautern.

Ullman, J.D. (1980). *Principles of Database Systems*. Computer Science Press, Potomac, Md.

Discussion

Question: Is near closer than close?

Answer: We ran some experimental tests and we got very few agreements between concepts and people. This really questions some of our perceptions about semantic regularity or uniformity. The meanings vary tremendously and we had trees of questions and answers regarding places on simple not complex maps. It was very surprising.

Question: Are those operations complete?

Answer: Yes. This is a complete minimal set. The operations are not overlapping each other. We can use this set for combining higher relations. For instance, disjoint and all the other relations are probably two groups. Disjoint has the definition such that two objects have nothing in common, while all the others require that at least some parts between the objects are in common.

Question: Why have disjoint? Just say that they do not meet.

Answer: We would have to say they are not intersecting, not meeting, etc. Not disjoint is equal to all of them.

Question: What does equality mean? You say that they are equal.

Answer: The equal relation means that two objects are the same. A house and the area it is built on are the same areas

Question: Why don't you use 'same'?

Answer: We should not discuss the selection of the words. This is not the object and this is a very difficult topic, and should be kept separate. It's up to you to select other words. What we want to present are the different types of relations.

Remark: One reason we are developing a standard query language is so that everyone will use the same words.

Question: What is the difference between intersect and overlap.

Answer: Intersection produces an object of a different dimension than the two objects. For example, two intersecting edges intersect in a node. Overlapping objects result in an object of the same dimension.

Question: It would be interesting to know how you can show that this set is a basic set. Because we have already shown that disjoint is not fundamental, that it is not necessary.

Answer: This holds only if you have a 'not'. If you do not have a 'not' then this is the simple set. The moment you have a 'not' you have one more operation and you can cancel one of them. But the 'not' counts toward the number of relations.

Question: Can you define the difference between an intersect and a meet?

Answer: Two objects meet if they have some common boundary. On the other hand, two objects intersect if they have only common inner parts.

Question: You discriminate between those by using the object type, because meet is most often associated with areas.

Answer: That's one of the things that all of these relations apply for any type of object, independent of the dimension. They apply between points, between lines, and between areas, but as well between a point and an area, between a line and an area, between a point and a line, and we can go up to three-dimensional space as well.

Question: What then is an overlap of a line?

Answer: An overlap of two lines is when one line starts with the other, runs some parts along the other, and finally ends outside the other line.

Remark: One thing you do need is a boundary operation.

Answer: You need a boundary operation in order to define operations.

Remark: You can define all of those operations in terms of boundary operations: boundary, co-boundary, and intersections.

Question: When you define the context, if you query the system say for 1984, will the context be coherent with the date too or just the information? Have you thought about that?

Answer: We are carefully excluding anything to do with time. We bite off quite big pieces so we know at least of some big bear traps. Everything that has time in it is one. There are bigger traps.

Remark: A lot of other people are interested in time .

CONCERNS REGARDING THE GEOMETRIC STRUCTURE OF THE NATIONAL DIGITAL TOPOGRAPHIC DATA BASE

David Armstrong
Topographic Engineering
Topographical Survey Division
Surveys and Mapping Branch
615 Booth Street
Rm. 755
Ottawa, Ontario
Canada
K1A 0E9

1. Introduction

The Topographical Survey Division of the Surveys and Mapping Branch (SMB), Energy, Mines and Resources Canada (EMR), is responsible for the development of the National Digital Topographic Data Base (NDTDB). Recently, the geometric structure of the NDTDB has been scrutinized regarding its ability to serve a community of users who wish to apply this data in a geographic information system (GIS) environment. The current state of the NDTDB and its future structuring directions will be discussed.

2. The Geometric Data Structure of the NDTDB

The NDTDB contains data which offers a numerical representation of the earth's topographic features from a national perspective at scales of 1:50 000 and 1:250 000. The 1:50 000 scale portion consists of three-dimensional data collected by digital stereocompilation methods. The 1:250,000 scale data is captured from existing map reproduction material using a combination of raster scanning and manual digitizing methods and is, hence, two dimensional.

The data exist as a collection of archival tapes organized on the basis of map sheet number. Each tape contains all of the topographic data pertaining to one map sheet. The 1:50 000 scale and 1:250 000 scale data are totally independent.

2.1 Current Structure Situation

The NDTDB presently conforms to the 'spaghetti' structure class which is a logical derivative of a digital data collection effort that focuses on graphic map production. At the root level, the position and shape of all features is defined in terms of two geometric primitives: points and lines. These primitives, however, from a geometric perspective, do not necessarily conform to a true logical representation of the terrain. This is especially true of data collected by stereocompilation, but also occurs in data digitized by scanning or manual methods.

For example, crossing lines need not be broken at their point of intersection. If they are broken, there will not necessarily be a mathematical connection between the end points of the lines which meet at the intersection. What results are small misregistrations called 'gaps' and 'overshoots'. Also, a linear element enclosing a polygon may not necessarily start and end at the same coordinate position, and linear elements representing a continuous feature, such as a stream, need not all run in the same direction.

Data existing in this form has been termed 'raw' data by the Division and is a result of digitizing convenience. With the sole intention of producing topographic maps, these data characteristics are not detrimental, however, if the NDTDB is to support GIS applications as well as map production, these characteristics inflict a serious drawback.

Recognizing this fact, the Division is striving to ascertain what the appropriate geometric structure is for the next generation of the NDTDB.

2.2 Future Structure Situation

The NDTDB must evolve into a standardized source of topographic information which better serves the digital data needs of a user community who are no longer limited to manipulation of the traditional map product. A major issue which faces the Division in this evolution is the level of detail and complexity which the geometric structure of the NDTDB must offer.

To classify different data structures, the Division has developed a hierarchy which reflects various levels at which a topographic data set may reside, see Fig. 1. As previously stated, the NDTDB currently consists of 'raw' data and resides at the base level of this hierarchy. For GIS applications this data has the fundamental shortcoming that it is void of implicit or explicit spatial relationships.

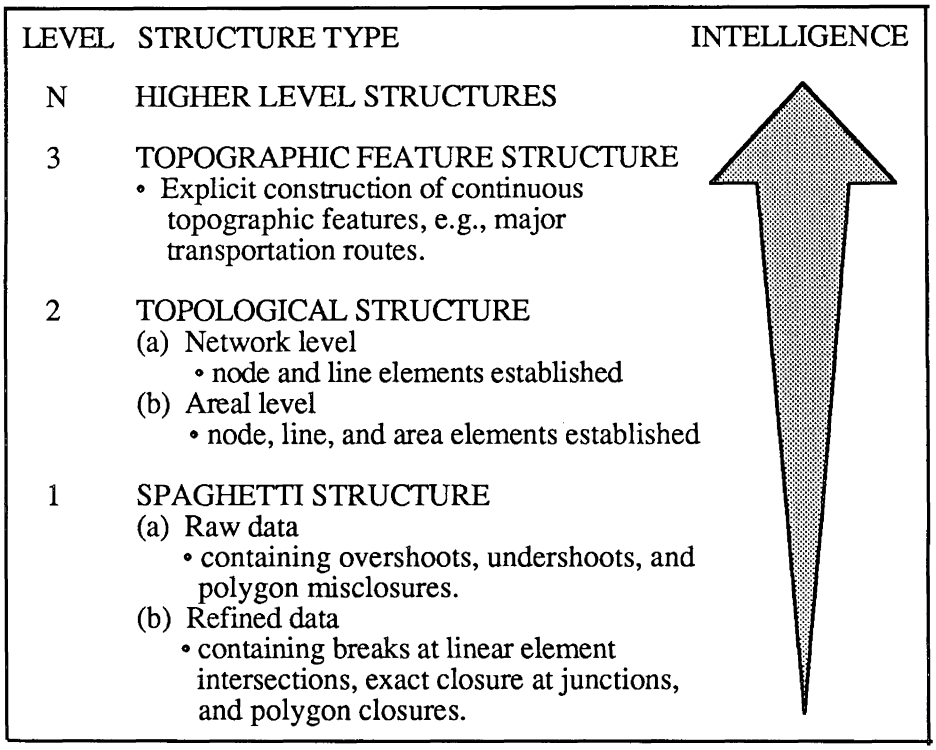


Fig. 1. Data structure hierarchy.

Implicit relationships such as connectivity and continuity could be satisfied if the NDTDB data is transformed to a 'refined' state by correcting spatial inconsistencies such as gaps, overshoots, and polygon misclosures.

Explicit encoding or spatial relationships such as connectivity and adjacency can be realized by employing a topological data model like that of the United States Geological Survey (USGS) [USGS, 1985]. More recently though, concern has been expressed that a higher level of feature recognition is required within topographic data structures,

[Guptill, 1986]. In Canada, an enhanced Digital Topographic Information Model (DTIM) has been developed by the Canadian Council on Surveying and Mapping (CCSM) which supports topology as well as the construction of continuous features from component topological elements [CCSM, 1986].

The Division is cognizant of the levels of geometric data structure complexity shown in Fig. 1 and have completed a strategic project to conceptually design the NDTDB. In addition, completed projects in NDTDB data structure enhancement include the creation of a topologically structured, 1:250 000 scale test data set of roadways and hydrography for distribution to other Federal departments, and the development of prototype software for the construction of topographic features (major routes) on the roadways theme.

The key determining factor in the structure complexity issue is feedback from users regarding their requirements. It is anticipated that the structured test data set will generate such feedback. In addition, recently acquired GISs present the opportunity to understand the input requirements of such systems and provide an environment within which cooperative projects requiring topographic base information may be undertaken.

3. Conclusions

The determination of the required level of geometric structure enhancement necessary for the NDTDB is key to its evolution. It is understood, however, that enhancement, whether it be to the refined, topological or topographic feature level, will place a burden on existing computer resources. The degree of this burden depends on the complexity of the structure level required and the methods used to implement this structure in the operational environment.

References

Canadian Council on Surveying and Mapping (1986). Proposed standards for a Digital Topographic Information Model. Draft Standards prepared by the Technical Subcommittee, Working Group 1, November.

Guptill, S.C. (1986). A new design for the U.S. Geological Survey's National Digital Cartographic Data Base. *Proceedings of Auto-Carto,* London, U.K., September.

USGS (1985). Digital line graphs from 1:24,000-scale maps. *United States Geological Survey Digital Cartographic Data Standards*, Geological Survey Circular 895-C, U.S. Geological Survey, Department of the Interior.

Discussion

Question: By refined data, do you imply that lines meeting at a node will bear the same nodal value?

Answer: Yes, assuming that your meaning of nodal value refers to the order of the node (i.e., the number of lines which connect at it).

Question: How much topology do you want to represent; the DLG or CCSM formats do not have a representation for all the topology ?

Answer: As a minimum, we want to support connectivity and adjacency relationships. The DLG and CCSM interchange formats provide for the explicit storage of these relationships.

Question: Are you aware that Harvard University has software to convert raw spaghetti data into a topological network.

Answer: No, but the Division has developed prototype structuring software in-house and has recently acquired CARIS and ARCINFO systems which are also capable of performing similar conversion.

Question: What is the advantage of transmitting topology?

Answer: It provides the spatial relationships necessary for geoprocessing. If the topology has to be reconstructed by the user, some inconsistencies may arise partly due to the limitation of individual computers.

Remark: This is not necessarily true as the data structure in the refined stage is sufficient enough to recreate the same topology if the user possesses this capability. Moreover, the data can be transmitted to integer format unlike the real numbering format of CCSM.

Question: How do you guarantee that the refined spaghetti data is correct unless the topological structure is established?

Answer: You have to built the topological level to ensure that the refined data is correct. There will be no topological inconsistency problems by distributing spaghetti data provided that the data are 'clean' or 'refined'. It can be very difficult to achieve the refined data level so it may be more feasible for users to receive raw spaghetti data and reconstruct the topology themselves.

Remark: But this will impose a lot of processing overhead on the topology users. Furthermore, the future market of digital data will be in geoprocessing rather than

computer cartography, hence it seems logical that producers should supply topographic data in a topological or refined state.

Remark: In utility applications, the prime usage of digital data is for graphical overlaying of different layers of information. i.e., a carbon copy of base maps with other information.

Remark: In a workshop organised in Alberta for the utility and oil companies, the feedback was that there was a strong demand for spatial data for geoprocessing apart from the carbon copy usage.

Remark: In Bell Canada, the main use of digital data is in the AM/FM (automated mapping facility management) graphic application.

Remark: It is hard to know the needs of users. Price factors can also influence it significantly. At the moment, the management of resources is expensive due to the limitations of computers and the high cost of software.

Remark: As for the present LIS of most oil companies, it is mainly comprised of some mapsheets filed together. But, with the availability of digital data from the survey department, they are investing in the infrastructure needs of spatial data base.

Remark: The status of digital output from EMR is indirectly controlled by the market, and it is advisable not to introduce a data format that is too advanced for the users.

Remark: LRIS (Land Registration and Information Service) conducted a user needs study on the possible usage of digital geodata in 1980, and it came to the conclusion that there was no use for it . Within two years later, however, there are agencies (timber industry) that have had to collect their own data as LRIS cannot cope with their pace of demand.

Remark: At present, the user has no sound knowledge of what the digital data can offer and is unable to provide a meaningful response until they have been exposed to it. The digital collection and exchange policy should be set up by the government as it is the infrastructure that drives the system.

Remark: Generally, the user does not know what they want and EMR is planning to set up a 'cooperative environment' within which the user can be exposed to the capabilities of digital data and hopefully its potential can be realized. It will be beneficial to the users when deciding to purchase a digital system in the future.

Question: Is the structure of the current digital data compatible to those of the future?

Answer: Once the refinement stage is achieved, it will be quite easy to transfer it to any other system.

Question: What do you mean by refinement of data?

Answer: Refined data is spaghetti data which contains no explicit encoding of topological relationships. It is, however, void of spatial inconsistencies such as gaps, overshoots, and polygon misclosures which result in an illogical representation of topographic reality. Spatial relationships are 'implicit' in refined data. Explicit relationships such as connectivity and adjacency can be readily derived by computation. There are basically two approaches to the refined data approach: (1) Refine data within themes, but not between themes. (2) Provide complete vertical integration of all the data set themes.

Remark: In the cleaning process, it is best to start with the 'overshoot' and 'intersection' problems for each layer and then finally proceed to the vertical integration of various layers.

Remark: The conversion of a clean topology from one format into another is a trivial procedure.

Remark: Another challenging problem is to maintain consistency of topology during data manipulation. Usually, the explicit topology has to be reconstructed from spaghetti data on every occasion. In the TIGRIS system, the topological consistency is being maintained at all times.

Question: What do you intend to do with the mapsheet boundary?

Answer: The cleaning procedure works on mapsheet basis and it does not involve cross sheet information, i.e., no topology at the borders. This is crucial as users may use different map sheet formats than those provided.

Question: Did you consider the prospect of giving values to the data according to its degree of usage?

Answer: The infrastructure information such as hydrographic or contour generation data are very important but there is no specific plan of implementing the 'usage' value yet.

Remark: The contour line is not a good representation of the terrain and USGS provides only gridded DTMs (digital terrain models) instead. The contour line has to be generated by the user and it saves on some cleaning up processes.

Remark: There should be cross checkings between the relationship of information from different layers. Also, a strong communication link among the organizations supporting the data should be established.

CURVE GENERATION TECHNIQUES FOR COMPUTER-AIDED CARTOGRAPHY

Y.C. Lee
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, New Brunswick
Canada
E3B 5A3
SE@UNBMVS1.bitnet

1. Introduction

Smooth curves form an important graphic component of a map. The cartographer is traditionally relied upon to generate accurate and aesthetically pleasing curves. Such skill is usually the result of considerable practice. With the introduction of computer-aided cartography (CAC), this task of curve generation is mainly performed by the computer.

The aesthetic quality of a line is related to its shape and smoothness. These qualities are not of pure cosmetic nature, but describe important mathematical properties of a line. For example, it is not enough that a highway on a map be only positionally accurate, the line representing the highway must also mimic its smooth curvature.

When the cartographer is reproducing a highway on a map by tracing its outline, he imposes on the line certain constraints he knows about the highway. Automatic curve generation is also itself a game of constraints. The difference here is perhaps in the way of expressing the constraints – mathematical quantities are used instead of intuition. The many different methods of curve generation differ mainly in the way shape and smoothness of a line is being controlled.

Smooth curve generation is useful to CAC in the following areas:

- Reproduce curves in nature: Many digital maps are digitized either from existing maps or stereo-models. The operator traces lines and the hardware generates discrete points along the line. Automatic curve generation methods are then used to interpolate between the digitized points to obtain a smooth line. This approach relies heavily on the curve generator's ability to reproduce the original shape faithfully without much user intervention.

- Design and edit curves: Some of the curves manually digitized contain errors which must be corrected interactively. The operator must have some means of controlling the shape of the resulting curve. These means of control must be intuitive and easy to use so as to minimize the number of iterations required to obtain the desired curve. Curve design is particularly important in thematic mapping where the cartographer must be able to design a curve to his exacting requirements.

- Compact and smooth data: In this case, the inverse of interpolation is involved. That is, given a line represented by many data points, what is its best approximation by a mathematical curve? Generally, the mathematical curve requires less space to store than the original line composed of many points. If this approximating curve can filter noises in the original curve, then smoothing can also be achieved.

This paper surveys the curve generation methods based on polynomials. Some fundamental concepts of curve generation will be introduced, followed by the development from simple polynomial curve generation methods to the different types of spline functions. The properties of each type and their applicability to CAC will be explained.

2. The Problem of Curve Generation

There are basically two approaches to curve generation: interpolation and approximation. **Interpolation** means the generation of a smooth curve passing through a number of given points (called **data points** or **control vertices**), and **approximation** means the generation of a smooth curve through the vicinity of the data points. In the case of interpolation, there are theoretically an infinite number of curves which can be generated through the data points. Practically, however, the curve must satisfy certain constraints such as shape and curvature.

The tendency for human beings to mentally 'fill in the gap' (the tendency to interpolate) between points has been studied in Gestalt psychology. What remains unknown is the mental algorithm used, and why the algorithms employed by different

persons are so similar. Because of this, interpolation by mathematical functions can only imitate with limited success this complex human process. This has led to the proliferation of interpolation methods invented to suit different applications.

The basic mathematical principle behind curve interpolation is that: If we know the behaviour of the curve at the data points, it is then possible to reconstruct the curve. The known behaviour of the curve, called the **constraint**, controls the shape and smoothness of the curve. Most of these constraints, such as first and second derivatives, are analytical in nature. Hence interpolation methods are very effective if we know the mathematical nature of the curve and its constraints at the data points. The methods are not as effective for interactive design because of the need to change the curve. In order to change the curve using interpolation techniques, we must modify the analytical constraints at the data points. The problems are that these contraints are too abstract for most users, and that the effect of these constraints on the modified curve is not always obvious. The transformation of these analytical constraints to geometric constraints led to the development of a new type of curve approximation methods.

Traditionally, curve approximation generates a curve which is close to the data points according to some mathematical condition such as minimizing the distance from the points to the eventual curve. Again, the condition used is too analytical and is difficult to manipulate in an interactive environment. The need to interactively design a curve has brought a new perspective to curve approximation. Many new approximation techniques have been developed to employ geometrical conditions. One of the most popular geometrical constraints used is the **shape polygon**. Curve approximation techniques based on the shape polygon will generate a curve within the convex hull of the shape polygon (Figure 1). Given a number of points, the **convex hull** is the polygon formed by stretching a piece of rubber band around the points. A curve generated by these techniques generally does not pass through all the data points. But due to the convex hull property, the curve follows the shape of the polygon defined by the data points. Moreover, the shape of the curve can be changed by modifying the shape polygon.

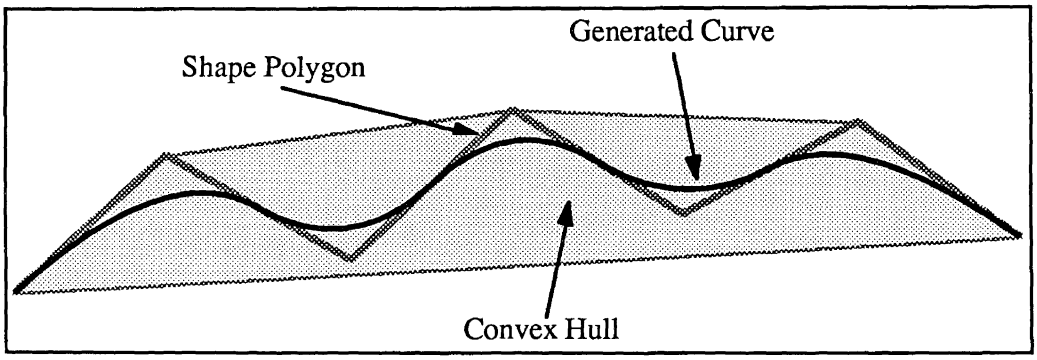


Figure 1. Curve within the convex hull of the shape polygon.

3. Curve Generation with Polynomials

Of all the mathematical functions used for curve generation, polynomials are the most popular because they are simple in form, easy to evaluate and differentiate. One way of writing a polynomial of degree $\leq n$ is to use the **standard** (or **monomial**) form:

$$P(x) = a_0 + a_1 x + \ldots + a_n x^n \qquad (1)$$

This, however, is not the only form. Other forms generate the same polynomial, and are chosen mainly for computational and practical reasons. The other forms will be introduced later in this paper.

Let us assume for the time being that the data points (x_i, y_i, i = 0, n) are single-valued. That is to say no two data points share the same x-coordinate. This is not a realistic condition in a mapping environment, but we will tackle the problem of representing multi-valued curves later.

An interpolating polynomial $P(x)$ is one that satisfies the following condition:

$$P(x_i) = y_i, \qquad i = 0, n \qquad (2)$$

where y_i is the value of the original function at x_i. In other words, the value of the interpolating function at each of the data points is that of the data point (x_i, y_i). The x_i are called the **knots** of the interpolating function. An approximating polynomial $P(x)$ is one such that

$$P(x_i) = y_i, \qquad i = 0, n \tag{3}$$

for which $d_i = | P(x_i) - y_i | > = 0$. For $P(x)$ to be of any practical use, the deviation d_i between the interpolated and the given values is small and is governed by some pre-defined criteria. A well-known criteria is to minimize d_i resulting in a least-squares approximation.

4. Lagrange Interpolation

A classic interpolation method is based on a well-known property of the polynomial: an n degree polynomial contains n+1 coefficients. Hence, given n+1 distinct points, it is always possible to find a polynomial of degree n passing through the points. The French mathematician Joseph Louis Lagrange (1736-1813) proposed a method to calculate such a function. He expressed the polynomial as

$$P(x) = \sum y_i L_i \qquad i = 0,n \tag{4}$$

which is a polynomial of degree n, and the L_i are the cardinal basis functions. A cardinal basis function L_i has a value of 1 at x_i and zero at all other x_i's. From this it is easy to see that

$$P(x_i) = y_i$$

which satisfies the interpolatory requirement. The solution of $P(x)$ then consists of the solution of the L_i's which are dependent on the data points.

The Lagrange interpolating polynomial is simple and elegant except for one major flaw: the resulting polynomial is of excessively high order. It is well known that a polynomial of degree n has potentially n roots, and hence it might cross the x-axis many times causing excessive undulations of the curve. A line in a map usually contains a large number of points.

The Lagrange form of the polynomial is only different in appearance from the standard form given in equation (1). They are mathematically the same polynomial and behave identically. If the other form were used, a different method which involves the

inversion of a big matrix would have been required to solve for all the a_i's. The point here is that the use of a different set of basic functions leads to a different method of solving the problem. The solution of the L_i's in the Lagrange form does not involve matrix inversion and is therefore more stable.

Another benefit of considering basic functions (and subsequently the computational aspect) of interpolating polynomials is to understand some properties of the function. A look at different methods of computing the same function can often reveal some properties which are obvious in one method but not in another. Take for example the standard form of the polynomial. Although it is not computationally desirable, this form is easier to understand and can illustrate one important property of a Lagrange interpolating function, which is its tendency to oscillate. The solution of equation (1) involves the solution of a linear system of equations, involves all the coefficients of the polynomial (a_i) and all the data points. Therefore, the mere change of one data point affects the entire system of equations, which affects the interpolating function. The interpolation method based on a polynomial which is affected by the change of a single point is termed **global**. This is not a desirable property for a curve that is to be edited interactively. No cartographer is happy with a scheme that affects the entire line when only a portion of the line is being changed.

For these reasons, the Lagrange interpolating polynomial is not of much practical use in CAC. However, it is a very basic interpolating polynomial which we will use to build the more complex schemes.

5. Piecewise Interpolation

An effective way of reducing the degree of the interpolating polynomial is to use many pieces of lower-order polynomials (called **segments**) joined together instead of a single function. They collectively form a piecewise function. The well-known method of joining every two data points with a straight line results in a piecewise liner interpolating function. Higher-order piecewise functions can be obtained by piecing together higher-order polynomials. For instance, a quadratic can be generated over every three points. Since each of the quadratics is in fact a Lagrange function, we will call this composite

function a piecewise quadratic Lagrange polynomial; Simpson's rule of calculating the area under a curve actually assumes such a piecewise function.

While solving the oscillation problem, the piecewise Lagrange function creates a smoothness problem: the interpolating curve is not smooth where the segments meet (called the **junction** or **joint**). Such a polynomial is said to be not differentiable over the entire range because it is not always differentiable at the joints. Note that a certain configuration of the data points could cause the piecewise Lagrange interpolating polynomial to be differentiable over the entire curve. This is due to the uniqueness of interpolating polynomials. That is to say, if four data points actually fall on a quadratic polynomial, a Lagrange interpolating function over the four points will reproduce the quadratic and not another cubic. Now suppose we are given eight data points which actually fall on a quadratic polynomial. A piecewise cubic Lagrange polynomial can be generated by a composition of two cubic Lagrange polynomials, one over the first four points, and another over the other four. By the property of uniqueness, each cubic is in fact a quadratic and the composite curve made up of the two quadratics matches exactly the original quadratic polynomial. In this case, the place where the two quadratics meet is perfectly smooth and is therefore differentiable. This, of course, is a highly contrived case just to illustrate the point that piecewise Lagrange polynomials can be smooth.

A piecewise Lagrange polynomial is no longer global since the change of one data point affects only one segment. Such an interpolating function is called a **local** scheme. It must be pointed out, however, that not all piecewise functions are local.

All the interpolating functions we will consider in this paper are piecewise. Various methods will be described to show how the degree of differentiability can be controlled at the joints.

Differentiability of a curve is related to the visual smoothness of the curve. The more times a curve is differentiable, the smoother it appears as illustrated in Figure 2.

Figure 2. Differentiability and smoothness (after de Boor [1978]).
The curves from left to right increase in differentiability.

6. Hermite Interpolation

This is a piecewise polynomial interpolation scheme attributed to Charles Hermite (1822-1901). The method requires the knowledge of some derivatives at the data points besides the position of the data points themselves. Take the case of the cubic Hermite polynomial. A cubic is used to interpolate every pair of data points. The four constraints necessary for solving each cubic are the two data points plus the slope at each of them. Each data point (p_i) now is the meeting place of two cubic segments, one to the left and another to the right. The left cubic is uniquely defined by p_{i-1}, p_i and the slopes p'_{i-1}, p'_i whereas the right cubic is uniquely defined by p_i, p_{i+1}, p'_i and p'_{i+1}. If the two cubics use the same slope at p_i, then the joint of the two cubics at p_i is smooth; the left piece ends and the right piece starts with the same slope. Since a curve so constrained is guaranteed to be differentiable at least once at the joints (the cubic themselves are differentiable indefinitely), a cubic Hermite interpolating polynomial is said to be once differentiable.

A cubic Hermite polynomial is local and is smooth. It seems to be an ideal choice for cartographic applications only if the slopes at all the data points can be somehow approximated. A number of methods to approximate the slopes have been proposed, each leading to a somewhat different curve. We will describe two of these methods.

6.1. Bessel's Approximation

This classic scheme approximates the slope at the point p_i by the slope at p_i of a quadratic passing through p_{i-1}, p_i, and p_{i+1} [Conte and de Boor 1972]. In order that the same scheme applies to the two end points, it is necessary to extrapolate two points, one at each end of the curve

6.2. Akima's Approximation

Akima's method [Akima 1970] uses five data points to estimate the slope: two points on both sides of the point in question. The slope at point p_i is the weighted mean of the difference in slopes on both sides.

The important property of Akima's slope estimation is that if three points are collinear, then a straight line will be generated through them. This is not always the case using Bessel's estimation. In general, Akima's method gives tighter curves and less inflections. Being a member of the piecewise cubic Hermite interpolation, Akima's interpolation method also produces a local, once-differentiable function.

7. Cubic Splines

A cubic spline, introduced by Schoenberg [1946], is a mathematical equivalent of the physical spline which consists of a flexible rod mounted on heavy weights called ducks. The rod can be bent to the desired shape by moving the ducks. The term spline is a generic name for all n-degree piecewise polynomials which are n-1 times differentiable, although splines other than cubic are not mathematical models of the physical spline.

A cubic spline is a piecewise polynomial which is twice differentiable over the entire range of the curve. This property gives a cubic spline a smoother look than that of a piecewise cubic Hermite polynomial.

The way to achieve twice differentiability at the joints is to set up a linear system of equations forcing this condition at all the data points. This set of equations, including all the data points, actually solves for the slopes at the data points so that the change of the slope (the second derivative) across each data point is continuous. This is just another way of obtaining the slope at the data points. The Hermite interpolating scheme can then be used to solve for the cubic spline polynomial.

Although the cubic spline is a piecewise polynomial, it is not a local function. If we recall the linear system of equations involved in finding the slopes at the data points, a change in any one point affects all the slopes and hence the shape of the entire curve. The effect of moving a data point, however, diminishes with distance.

It turns out that two extra conditions, other than the position of all the data points, are required to uniquely solve the system of equations used to find the slopes at the data points. A number of choices are possible. The most common choice is to force the second derivatives at the two end points to be zero. Using the analogy of the physical spline, this is equivalent to releasing tension at the ends of the spline. Such a spline is called the natural cubic spline. Another choice is to force the slope at the first point to agree with that of the last point. This results in a **periodic spline** which is useful for closed loops.

The global nature of the cubic spline is more a computational problem than a shape control problem because the effect of changing one data point is dampened to an insignificant degree a number of segments away. The number of segments required to dampen this effect increases with the degree of the spline and its global effect will become more pronounced. If the degree is high enough, the spline becomes the Lagrange polynomial.

The computation problem involved with the global nature of the cubic spline, however, can be annoying since each single change to the curve involves re-computation

of the spline function. This makes the cubic spline not a very attractive function or curve design.

Another undesirable effect of the cubic splines is its tendency to produce extraneous oscillations (Figure 3). This is because the cubic spline must maintain a smooth change in slope over the entire curve and hence cannot change direction quickly. This inability to turn tight corners is a price one pays to obtain a smoother looking curve. Attempts have been made to control the oscillations. One means of control is to apply tension at the data points resulting in **splines under tension**, a scheme originally developed by Schweikert [1966]. His formulation involves hyperbolic functions such as sin$h$, and is computationally complex. A polynomial alternative, called the v-spline, was developed by Nielson [1974].

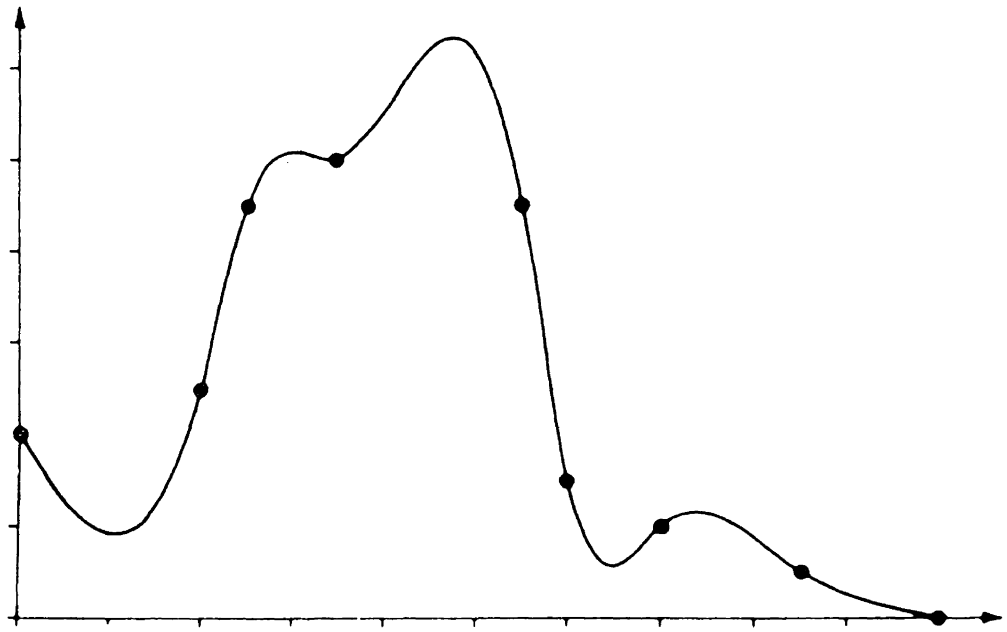


Figure 3: Extraneous oscillations on a cubic spline (after Späth [1974]).

Despite these shortcomings, the cubic spline has been extensively used and intensively studied over the last three decades since its invention. Unlike most of the Hermite type interpolation methods which are ad hoc in nature, the cubic spline is

rigorous. It also leads to a very powerful curve generation tool, the *B*-spline, which we will discuss later.

8. Parametric Interpolation

So far we have restricted our interpolating function to be single-valued, that is, no two data points have the same x-coordinate. This is not realistic in mapping applications since most lines on a map wander freely and some even form closed loops.

An effective method of overcoming the restriction of equation (1), which is also called an implicit polynomial, is to convert it to the parametric form by dividing that simple polynomial into two, both a function of a new parameter u. One of the parametric equations expresses the change in the x-coordinates against u (the knots), while the other expresses the change in the y-coordinates against u. This *divide and conquer* method expresses the change in x and in y by two single-valued functions, and their combined effect allows the curve to be multi-valued. An example of a parametric polynomial is

$$\begin{aligned} X(u) &= au^2 \\ Y(u) &= au \end{aligned} \tag{5}$$

which represent the parabola $y^2 = 4ax$. In general, a parametric polynomial is denoted as

$$\boldsymbol{Q}(u) = (X(u), Y(u)) \tag{6}$$

which is called a **vector function**. Note that both $X(u)$ and $Y(u)$ are piecewise and hence composed of many curve segments.

The choice of the parameter u, however, can affect the shape (and hence the smoothness) of the curve. Obviously, one important requirement of u is that it increases monotonically so that each of the parametric polynomials are single-valued. The simple set 1, 2, 3, ..., n could serve as values of u and the collection (u_0, u_1, u_2, ..., u_n) forms a **knot vector**. An empirically better choice is the accumulated chord length which is the accumulated length of the polygonal line passing through the data points. The

optimal parameterisation, however, is the arc length. Unfortunately, the arc length is not normally known before interpolation is complete.

For a piecewise parametric polynomial, the knots divide the curve into segments. Within each segment, a local parameter t normalized to range from 0 to 1 is often used. This is to enable uniform description of these segments. Hence the i^{th} segment of the piecewise function is denoted by

$$\boldsymbol{Q}_i(t) = (X_i(t), Y_i(t)) \tag{7}$$

in which each of X_i and Y_i is a function of t.

The differentiation of a vector-valued function is performed by differentiation of the two component functions. In other words, the first and second derivatives of $\boldsymbol{Q}_i(t)$ are expressed as

$$\begin{aligned} &\boldsymbol{Q}_i'(t) = (X_i'(t), Y_i'(t)), \text{ and} \\ &\boldsymbol{Q}_i''(t) = (X_i''(t), Y_i''(t)). \end{aligned} \tag{8}$$

Hence the continuity of $\boldsymbol{Q}_i(t)$ across the segments of the piecewise function can be controlled by continuities of $X_i(t)$ and $Y_i(t)$. In particular, $\boldsymbol{Q}(u)$ becomes a cubic spline if we impose continuity of the first and second derivatives of $X(u)$ and $Y(u)$ which are themselves cubic functions. That is to say, $\boldsymbol{Q}(u)$ is a cubic spline if $X(u)$ and $Y(u)$ are cubic splines. This gives us a way of constructing the parametric cubic spline $\boldsymbol{Q}(u)$.

It is very easy to convert any of the interpolation functions discussed so far to the parametric form. It simply involves picking an appropriate t, changing the original parameter to t, and solving two equations instead of one.

We can observe some properties concerning the continuity of parametric polynomials. Consider two neighbouring segments in $X(u)$ and $Y(u)$. If the composite curve is once differentiable across the segments, then $X'_i(1)=X_{i+1}(0)$ or the first derivative of X with respect to t at the end of segment is identical to that at the beginning of the next segment. The same holds for function Y, namely, $Y'_i(1)=Y'_{i+1}(0)$. In other words,

$$(X'_i(1), Y'_i(1)) = (X_{i+1}(0), Y'_{i+1}(0)) \tag{9}$$

meaning that the **tangent vectors** at both sides of the knot are identical in direction and magnitude.

A continuous polynomial with its first derivative also continuous everywhere along the curve is said to have first-degree **parametric continuity** C^1. It is also called once differentiable. A cubic Hermite polynomial is one such function. If a curve, in addition to satisfying the above condition, also possesses continuous second derivatives it is said to have second-degree continuity C^2. It is also called twice differentiable. A cubic spline is one such polynomial. For a piecewise polynomial, we are particularly interested in the differentiability of the curve at the junction points because the differentiability at these points will ultimately determine that of the entire curve. For example, these junction points are only once differentiable in the cubic Hermite polynomial, whereas the curves between junctions, which are cubics, are infinitely differentiable.

When a piecewise cubic curve is once differentiable, the tangent vectors at both sides of the junction points are equal in slope and magnitude. However, this is not a necessary condition for a curve to appear smooth. For a curve to be **visually (geometrically) smooth**, it is only necessary for these tangent vectors to be equal in slope. Since this is the continuity actually seen, it is called the **geometric continuity** G^1. An example illustrating the concept of geometric continuity is given by DeRose and Barsky [1988]. This concept of geometric continuity can be carried to a higher order. A piecewise polynomial is said to be geometrically continuous up to the second derivative, or G^2, if in addition to being G^1, the curvature vectors are also continuous across the junction.

9. Rational Polynomial

The conic sections, except the parabola, cannot be represented by an ordinary polynomial. That means ellipses, circles, and hyperbolae cannot be generated by the interpolation schemes discussed so far. However, the conic sections can be produced by a projective transformation of the parabola using the following expression:

$$f(t) = \frac{a_0 + a_1 t^1 + a_2 t^2}{1 + t^1 + t^2} \quad . \tag{10}$$

This form of expression with polynomials as numerator and denominator, is called a **rational polynomial**. A circular arc can be generated by the following rational parametric polynomials:

$$x = \frac{1 - t^2}{1 + t^2} \quad ,$$

$$y = \frac{2t}{1 + t^2} \quad . \tag{11}$$

10. Curve Approximation for Interactive Design

Most of the interpolation schemes used today are based on piecewise cubic polynomials. In other words, the different schemes are basically different methods of computing a cubic polynomial. The computational differences among the various methods affect not only the shape of polynomial produced, but also how the user can change the shape.

It is obvious that the shape of the polynomial can be changed by moving the data points. Depending on the method used, the effect of moving a point is different. Using a cubic spline, the modification of a point affects the entire curve, whereas when using a Hermite cubic polynomial the effect is localized. This is a highly desirable property for interactive curve design.

The position of the data points is only one of the contraints used to create a cubic. The other contraints are the first derivatives at the points. Using the Hermite interpolation method, the first derivatives are given or approximated. The cubic spline method uses a purely analytical approach to determine the first derivates by enforcing the condition that the second derivates are continuous at all the knots. Hence, if the Hermite scheme is employed, the user can further modify the shape of the curve by adjusting the slope of the curve at the data points. This is not possible if a cubic spline were used to interpolate the points. The reason for this is simple: if the first derivatives were to be changed by the user freely, then the second derivatives cannot be guaranteed to be continuous, and hence the polynomial can no longer remain as a cubic spline.

Let us explore further the possibility of using a Hermite interpolating polynomial for interactive curve design. Recall that in a parametric case, the first derivative at a data point is represented by a tangent vector. After a data point has been fixed, the curve can still be changed by adjusting the direction and magnitude of this vector. This provides a geometric (or visual) tool for the user to adjust the shape of the curve.

Despite the fact that these tangent vectors are rather long compared to the curve, they are fairly effective tools, particularly in an interactive environment in which the effect of the change is immediately visible.

The problem of excessive length of the tangent vectors can be solved by scaling them. This leads to a very useful scheme, known as the Bézier cubic interpolation, for curve approximation. In this method, the tangent vectors are one-third the size of the true vectors.

10.1. Bézier Cubic Interpolation

Scaling the tangent vectors to one-third of their size produces another interesting property. Each tangent vector has two points, one on the curve and another away from it. Hence each curve segment has two of these vectors, one at both ends of the curve. The four points on the two vectors define a quadrilateral, and the curve lies entirely within this four-sided polygon. These polygons are called **shape polygons** because they approximate the shape of the curves.

The shape polygons are useful both for defining the curve in the first place and for modifying the curve later. To start the design, the cartographer would determine the data points which form the shape polygon and approximate the eventual curve. After the curve has been generated, the cartographer can then move the data points (the vertices of the shape polygon) to adjust the shape of the curve. It is helpful to point out that the moving of the data points is equivalent to changing the magnitude and direction of the tangent vectors.

10.2. B-Splines

B- (or basis-) splines are splines from which other splines can be constructed. We have illustrated earlier that a polynomial can be expressed in terms of various types of basis functions. We have discussed the cubic spline expressed as the linear combination of the set $(x^3, x^2, x, 1)$. The Bézier polynomial is constructed from Berstein basis. The B-splines are but another type of basis functions for spline polynomials.

Since B-splines are themselves splines, they are piecewise polynomials of a certain degree. Cubic splines are built with B-splines of the third degree and quadratic splines with B-splines of the second degree.

B-splines are very special splines: they have a value of other than zero only over a limited number of intervals. In particular, a cubic B-spline is non-zero only over four intervals. The polynomial within each interval, naturally, is a cubic.

Using B-splines, a cubic spline can be expressed as:

$$S(x) = \sum a_i B_i(x) \tag{12}$$

where B_i are the B-splines of the 3rd degree. In order to make this an interpolating spline, we require that

$$F(x_j) = \sum a_i B_i(x_j) \tag{13}$$

where the only unknowns are the coefficients a_i. Cox [1971] and de Boor [1972] proposed a recursive algorithm to evaluate the B_i at each x_j. After this, the coefficients a_i can then be solved by a set of linear equations.

The interesting observation, however, is that the coefficients a_i in equation (12) follow closely the shape of the cubic spline, and act much like the shape polygon in the Bézier polynomial. By not solving for the a_i's in equation (12), but by replacing them with data points, we obtain a parametric representation of an approximating cubic spline:

$$s_x(t) = \sum x_i B_i(t)$$

$$s_y(t) = \sum y_i B_i(t) \quad . \tag{14}$$

The parameter t is often chosen to be integers running from 0 to n, the number of data points. The basis functions B_i in this context behave like a blender in that they mix the effect of the two data points at the end of the each cubic piece and distribute their influence over that portion of the curve. The B_i's are hence sometimes also called **blending** or **weighting functions**.

The approximating B-spline possesses the advantages offered by the Bézier polynomial. In fact, the general B-spline includes the Bézier curve as a special case. If an n-1 degree B-spline is generated with n data points, the resulting curve is also a Bézier curve.

The following is a list of interesting properties about the cubic B-spline.

- Like the Bézier curve, multiple data points reduce the differentiability of the curve. For example, three coincidental control points force the cubic-spline to bend sharply at the point. A similar effect can be obtained by multiple knots. This reduction of differentiability somewhat violates the definition of a spline. They are hence sometimes called **sub-splines**.
- Each point on the curve is influenced by only four data points. This makes the cubic B-spline a local interpolation polynomial.
- Like the Bézier method it processes the convex hull property, but much stronger: each cubic-spline piece lies within the convex hull of the four control points influencing it. That is to say, the cubic B-spline follows even more closely the shape of the shape polygon (Figure 5).
- If four control points lie on a straight line, their associated curve is also a straight line.

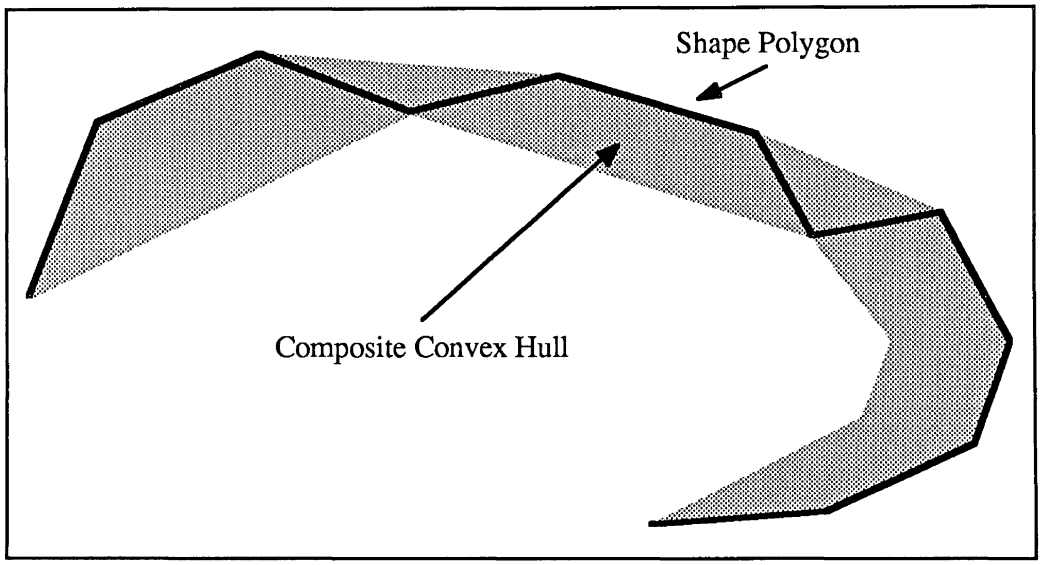


Figure 5. Convex hull property for a cubic B-spline.

The B-spline shows many of the advantages of the Bézier curve. In fact, it includes the later as a special case. This makes the B-splines, particularly the cubic B-splines, a widely used tool in curve generation.

10.3. Beta-Spline

Traditionally, the only way to change the shape of a spline is to move its data points. The realization that geometric continuity (G^n) is less restrictive than parametric continuity (C^n) leads to Beta-splines which allow extra shape controls. In other words, a G^2 curve is not necessary of C^2 and the same holds for G^1 and C^1. Hence, a geometrically continuous curve has additional degrees of freedom left in the formulation to be used for extra control. The Beta-spline [Barsky 1981] is uses the two additional degrees of freedom to control the shape of a spline. The Beta2-spline [Barsky and DeRose 1983] retains the C^1 continuity but uses G^2 continuities and uses the one additional degree of freedom to control the global shape. Piegl [1987] applies a similar approach to produce G^1 Bézier curves.

11. Conclusions

Mathematical curve generation has not been a great concern for digital mapping. There are a number of reasons for this. A very dense sampling rate in map digitizing has been the norm. For very closely spaced points, the choice of an interpolating function is not of prime importance because a linear one is almost as good as any other. Besides, digitizing has been done mostly from existing maps where lines are already smooth.

The collecting and editing of digital data has been mainly batch-oriented in the early days of digital mapping. Editing is done not by changing the old line, but by removing it entirely and the new line redigitized. In some older systems, editing can only be done manually on the hard copy.

Interactive editing has replaced batch processing and it is about time for a true computer-aided cartographic design system to emerge. The need for an approximating scheme is becoming more acute. The word 'design' is emphasized here to bring out the heavy man-machine interaction in such a system where the graphic elements can be reshaped at will by the designer with ease. This convenience can only be attained by allowing the designer to express his graphic requirements to the system in an intuitive manner.

It is true that one cannot totally replace the interpolating schemes with the approximating schemes. The two schemes serve different purposes and are both useful in curve design. Interpolating schemes are more useful in the initial input stage, whereas approximating schemes are easier to use while editing. A feasible approach is to allow easy conversion from one scheme to another. One such conversion method is outlined by Barsky and Thomas [1981]. Using their method, the user first digitizes a number of control points. An interpolating spline is then generated over the points. Finally the interpolating spline is transformed to an approximating one which the user can then reshape by moving the vertices of the shape polygon automatically generated by the program.

Variations of the cubic spline have been steadily growing over the years. The main objective was to discover better methods of controlling the shape of the curve in an

interactive environment. In other words, there were attempts to provide a more friendly user interface for curve design. A significant contribution was the application of geometric continuity in the generalization of interpolating functions. Beta-spline is the result of such application. It is anticipated that research in this area will continue, and more curve approximation techniques will be used in the field of computer-aided cartography.

Bibliography

Adams, J.A. (1974). Cubic spline curve fitting with controlled end conditions. *Computer Aided Design*, 6, 2-9.

Ahlberg, J.H. (1970). Spline approximation and computer aided design. In *Advances in Computers*, 10, Academic Press.

Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the Association for Computing Machinery*, 17(4), 589-602.

Barsky, B.A. (1981). *The Beta-spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures*, Ph.D. dissertation, Department of Computer Science, University of Utah, Salt Lake City, Utah.

Barsky, B.A. and S.W. Thomas (1981). TRANSPLINE — a system for representing curves using transformations among four spline formulations. *Computer Journal*, 24(3), 271-277.

Barsky, B.A. and T.D. DeRose (1985). The Beta2-spline: A special case of the beta-spline curve and surface representation. *IEEE Computer Graphics and Applications*, 5(9), 46-58.

Bézier, P.E. (1971). Examples of an existing system in the motor industry: the Unisurf system. *Proc. Roy. Soc.*, A231, 207-218.

Boehm, W. (1982). On cubics: A survey. *Computer Graphics and Image Processing*, 19, 201-226.

Brodlie, K.W. (1980). A review of methods for curve and function drawing. In *Mathematical Methods in Computer Graphics and Design*, ed. K.W. Brodlie, Academic Press.

Catmull, E.E. and R.J. Rom (1974). A class of local interpolating splines. In *Computer Aided Geometric Design*, eds., R.E. Barnhill and R.F. Riesenfeld, Academic Press, pp. 317-326.

Conte, S.D. and C. de Boor (1972). *Elementary Numerical Analysis*. McGraw-Hill.

Cox, M.G. (1971). The numerical evaluation of B-splines. *National Physical Laboratory DNAC 4*, August.

Cox, M.G. (1975). An algorithm for spline interpolation. *Journal. Inst. Maths. Applics.*, 15, 95-108.

de Boor, C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, 6, 50-62.

de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag, New York.

DeRose, T.D. and B.A. Barsky (1988). Geometric continuity, shape parameters, and geometric constructions for Catmull-Rom splines. *ACM Transactions on Graphics*, 7(1), 1-41.

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

Gasson, P.C. (1983). *Geometry of Spatial Forms*. Ellis Horwood, Chichester.

Gordon, W.J. and R.F. Riesenfeld (1974). B-spline curves and surfaces. In *Computer Aided Geometric Design*, eds., R.E. Barnhill and R.F. Riesenfeld, Academic Press.

Lancaster P. and K. Salkauskas (1977). *A Survey of Curve and Surface Fitting*. Published by the authors at 3052 Conard Drive, Calgary, Alberta, Canada T2L 1B4.

Manning, J.R. (1974). Continuity conditions for spline curves. *Computer Journal*, 17, 181-186.

McLain, D.H. (1978). Algorithm 100 — Vector approximation to curves. *Computer Journal*, 21, 178-180.

Nielson, G.M. (1974). Some piecewise polynomial alternatives to splines under tension. In *Computer Aided Geometric Design*, eds. R.E. Barnhill and R.F. Riesenfeld, Academic Press, New York, pp. 209-235.

Nowacki, H. (1980). Curve and surface generation and fairing. In *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, eds., G. Goos and J. Hartmanis, Springer-Verlag, Berlin.

Piegl, L. (1987). Interactive data interpolation by rational Bézier curves. *IEEE Computer Graphics and Applications*, 7(4), April, 45-58.

Pratt, V. (1985). Techniques for conic splines. *SIGGRAPH '85 Conference Proceedings*, 19(3), 151-159

Pravlidis, T. (1983). Curve fitting with conic splines. *ACM Trans. on Graphics*, 2(1), 1-31, January.

Prenter, P.M. (1975). *Splines and Variational Methods*. Wiley, New York.

Riesenfeld, R.F. (1973). Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design. Report Number UTEC-CSC-73-126, University of Utah, Computer Science Department, Salt Lake City, Utah.

Schoenberg, I.J. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4, 45-49, and 112-141.

Schweikert, D.G. (1966). An interpolation curve using a spline in tension. *J. Math. and Physics*, 45, pp. 312-317.

Schumaker, L.L (1981). *Spline Functions: Basic Theory*. John Wiley, New York.

Späth, H. (1974). *Spline Algorithms for Curves and Surfaces*. Utilitas Mathematics Publishing Inc., Winnipeg.

Discussion

Question: How do the shape parameters affect the Beta-spline?

Answer: There are two shape parameters β_1 and β_2. In the original design, they are global parameters but there were attempts to localize their effects. The second parameter β_2 is more symmetric than the first one, and is therefore more useful in an interactive environment. It behaves like a tension parameter and is therefore called tension in the Beta2-spline [Barsky and DeRose 1985] scheme which is a version of the Beta-spline providing only β_2 control.

Question: Is what they called the spline under tension the same thing as the Beta-spline?

Answer: The Beta-spline does provide a tension parameter but the name spline under tension refers to the scheme developed by Schweikert [1966] which is not based on the polynomial.

Question: You say that there is a transition from the cubic spline, the B-spline, to the Beta-spline. But isn't it more a requirement you put on your points? For the cubic spline you require the curve to go through the points, and with the other you just want a smooth curve that is close to the original points. So these considerations are fairly important when we design a curve.

Answer: There is a definite need for both interpolating and approximating schemes in cartography. The interpolating scheme is generally useful when the curve is digitized the first time. The approximating scheme is useful when changing the shape of the digitized line for accuracy or cosmetic reasons. Since most of the popular curve generation schemes are based on the cubic polynomial, it is possible to convert from one scheme to another. One such attempt was described by Barsky and Thomas [1981]. Catmull and Rom [1974], on the other hand, have developed a class of splines that can either interpolate or approximate the data points. They have since been called the Catmull-Rom splines.

Question: Is it better to use Akima's method instead of the cubic spline to smooth contours?

Answer: There is always the danger of having contours crossing after they are smoothed. In general, Akima's method would produce a curve with less oscillations than a cubic spline, and is therefore less dangerous.

Comment: I generally do not believe that any automatic curve generation technique should be applied to contour lines. You will immediately have contour lines intersecting.

Question: As soon as you add an extra feature (contour smoothing) on top of your model (the digital terrain representation) you are in a dangerous position, which is I suppose what you are saying. After smoothing, your contours no longer represent your original model. There is hence no longer any hope of reversibility, that is getting back to where you came from. So as soon as you want to clean up a contour in that fashion, you've got problems, conceptually anyway to my mind. However, this does bring up a question. This is an interesting problem because you have topological relations between contours — they fall inside each other and so on. We also have some nice topological relationships with respect to B-splines in that they have the convex hull property. Can you guarantee that if you use a B-spline or a similar function on contours which are concentric within each other, the resulting contours will not cross?

Answer: There are a number of curve generation methods, such as the Bézier curve and the B-spline, that shows the convex hull property. It is possible to check if the convex hulls overlap before generating the contours. If they are disjoint, then the smooth contours generated will not intersect. However, if they overlap, then there is very little we can tell. I am not sure how useful this test is in the practical sense because

the closer the contours the higher is the risk of their intersecting each other. This is where the convex hulls will likely overlap.

Comment: I think it is fundamentally wrong to smooth contours. All smoothing should be performed on the three-dimensional model instead and there are extensions of splines and other schemes to the third dimension. Is it possible to obtain spline-like descriptions of the contours directly from surface models such as Coon's patches?

Comment: I suppose my answer is that there are other methods not based on polynomials: the method of weighted average can serve much of the same purposes. Perhaps in some cases they are more manipulable. I was just concerned that no matter what theoretical implications are, there is always a need to smooth contours. It would be nice if we could guarantee that interpolated contours will not intersect given certain initial conditions.

Question: How difficult is it to extend curve generation methods to three-dimensional surface generation?

Answer: A convenient method of transforming a two-dimensional curve to a three-dimensional surface is to use tensor products. Suppose a spline curve is represented by

$$S(x) = \sum a_i B_i(x) \ .$$

The shape of this curve can be changed by modifying the coefficients a_i. That is to say, we can let $S(x)$ sweep out a surface simply by modifying a_i in the y direction. This can be conveniently done by defining the change of a_i with respect to y using another spline:

$$a_i(y) = \sum b_{i,j} B_j(y)$$

The resulting bi-cubic spline surface $S(x,y)$ is represented by the tensor product

$$S(x,y) = \sum_i \sum_j b_{i,j} B_i(x) B_j(y) \ .$$

This method can be applied to other interpolating methods to obtain tensor product Bézier surfaces and so on. However, the method is useful only for regularly spaced points.

Question: There is a scheme to represent splines using homogeneous coordinates. There are at least a couple of papers I have seen using that method. What do you think of them?

Answer: The homogeneous form is a convenient way of expressing rational polynomials.

Question: A concept that interested me lately is a hierarchical order of objects. In mapping, the aim is to produce not just one but several maps of different scales. There is a necessity to go from large scale to small scale and vice versa. Is there a way of representing splines in a hierarchical manner to facilitate scale changes?

Answer: This is basically the issue of generalization using splines. The problem has always been scale reduction. Enlargement is equivalent to backtracking in the hierarchy tree of scale changes, which is easier. Mathematical curves can help generalization in two ways. First of all, a mathematical definition of a curve is richer that the simple collection of coordinates. For example, we can derive derivatives easily from the function, and we can locate the points of inflection mathematically. These might help us to identify the characteristic points along a curve. Secondly, it might be possible to express generalization as a tension parameter. A curve at a larger scale is 'tenser' (or oscillates more) than one at a smaller scale.

Question: Is it possible to transform splines to the frequency domain? The nice thing about the Fourier transform is that the transformation is reversible. Many of the ideas of generalization seem to be related to the filtering of high frequency band widths. Are there easy ways of converting a spline representation to a frequency representation so that we can apply the Fourier transform?

Comment: I am not sure if it applies to piecewise functions, that seems to be the first question, and the answer is yes then how does it apply to these specific piecewise functions.

Comment: This can be done by sampling the curve.

Comment: No, it is not sampling. If it was sampling then we know how it works.

Comment: What is normally done for filtering is to perform convolution in the time domain, not in the frequency domain. I've seen papers about convolutions on curves in the spatial domain. It is completely analogous to the convolution of a time signal.

Comment: Convolution is expensive and that is why people use the Douglas-Poiker algorithm or similar methods. They are computationally less expensive.

Comment: The Douglas-Poiker method works very well for scale changes within a limited range. What we are aiming at are scale changes over a very wide range. I think the frequency domain is where one can probably remove high frequencies. The Douglas-Poiker method does not necessarily remove high frequencies.

POINT AND AREA INTERPOLATION AND THE DIGITAL TERRAIN MODEL

Christopher M. Gold
Memorial University of Newfoundland
St. John's, Newfoundland,
Canada
A1B 3X9
EMAIL: CGOLD@MUN.bitnet

Introduction

The purpose of this article is to provide a review of traditional interpolating techniques, together with their applications and weakness, and to discuss a 2-D analogue of 1-D interpolation that eliminates most of the problems and opens up the possibility of defining 2-D interpolation to match specific user requirements.

As discussed in Gold [1984], 'contouring' techniques have frequently confused issues of database storage and retrieval, sub-sampling site selection, neighbouring point selection, surface estimation procedures, and display issues. Due to a preoccupation with 'technique', for which any individual example may be more or less appropriate, the underlying objectives of the problem to be solved are often overlooked. In particular, the user frequently is of the impression that his data is precise, relatively sparse across the map sheet, and arbitrarily (and anisotropically) distributed — and that the program will adequately service these assumptions, providing an interpolated surface that precisely honours all data point elevations. The programmer, however, is often assuming (or hoping) that the data will be relatively evenly distributed, that a 'reasonable' density of data is available, and that 'minor' discrepancies between observations and the interpolated surface will not be significant. While this is certainly not the only scenario, it undoubtedly is only too common.

Clearly this scenario reflects an accident looking for a place to happen, based on the unspecified objectives of both user and programmer. If we accept the user's objectives as specified above, how do 'traditional' interpolation procedures perform?

Part 1. Traditional Methods

Again from Gold [1984], the most common traditional technique is to specify a grid over the map area and estimate elevation values at each grid prior to stringing contours through the grid. Thus, in general, elevation values of data points will not coincide with grid nodes, and the surface will be 'smoothed' to an extent depending on grid size. The most common method of estimating grid node elevations is some form of weighted average of the 'neighbouring' data points. Various other techniques are also used, in particular the generation of polynomial patches that fit the data of a small part of a map to some particular degree of accuracy, and are mathematically constrained to have a specified level of continuity between adjacent patches. A particular weakness of the patch approach is that in some systems the patch only approximates the data in that region. For all patch methods, however, the mathematical constraints to make adjacent patches fit together inevitably reduces the degree of continuity along patch boundaries — and this (usually slope continuity and curvature discontinuity) remains surprisingly visible on the resulting map, producing unacceptable results. For weighted-average techniques, the basic difficulty is in the compatibility of two separate steps — the selection of the adjacent data point to be used, and the range of influence of the weighting function applied to each of these neighbours [Gold, 1984]. It is particularly difficult to ensure that a data point's influence (weighting) has decreased to zero before it is discarded from the neighbourhood set; if this condition is not met, discontinuities in the surface will occur.

Lastly, as more types of data become available to a digital mapping system there is a strong need to be able to distinguish between, and handle appropriately, information that comes from a point source and information that pertains to a whole region or area.

Part 2. Design of Solution

The Missing Census Data Problem

Of the issues addressed in the previous section, the most significant appear to be the difficulty of always honouring all data point values, and the confusion between point and area observations. The first of these issues led to the necessity of using an adaptive methodology describing the relative locations of adjacent data points. The second issue required that point and area observations be treated alike.

This second point of usually addressed anyhow — by treating all area observations as point observations. For reasons to be discussed further below, the dual viewpoint appears to be more valid: all data points are expanded to areas. These areas, when expanded to meet the areas generated by adjacent point, would form the well-known proximal map (or Thiessen polygons, or Voronoi diagram) of the generating data points. There would thus be no functional difference between this diagram, a map of the coterminous United States, or a set of regular (square or hexagonal) grid cells formed implicitly by a regular grid or raster set of point observations. Thus subsequent processing could be handled consistently for all (or mixed) data types.

The value of this view was enhanced by a question posed by Tobler (originally in an invited talk in 1982, but subsequently published in Tobler and Kennedy [1986]). He was concerned with the problem of estimating the value for a single census area whose observation had been spoiled for some reason. He illustrated with a map showing the state of Kansas (Fig. 1a). If the value (e.g., cars per household) had been invalidated for some reason, how best could it be estimated from the values of the surrounding states? His suggestion was to use a weighted average of the surrounding values, with the weights being proportional to the length of the common boundary between that state and Kansas.

Examination of this technique showed that the common boundary length could easily be a very poor estimator of an adjacent state's influence; for example, if the common boundary was highly convoluted by comparison with the other relevant boundaries. A more appealing concept was initially to assume that the region under question did not

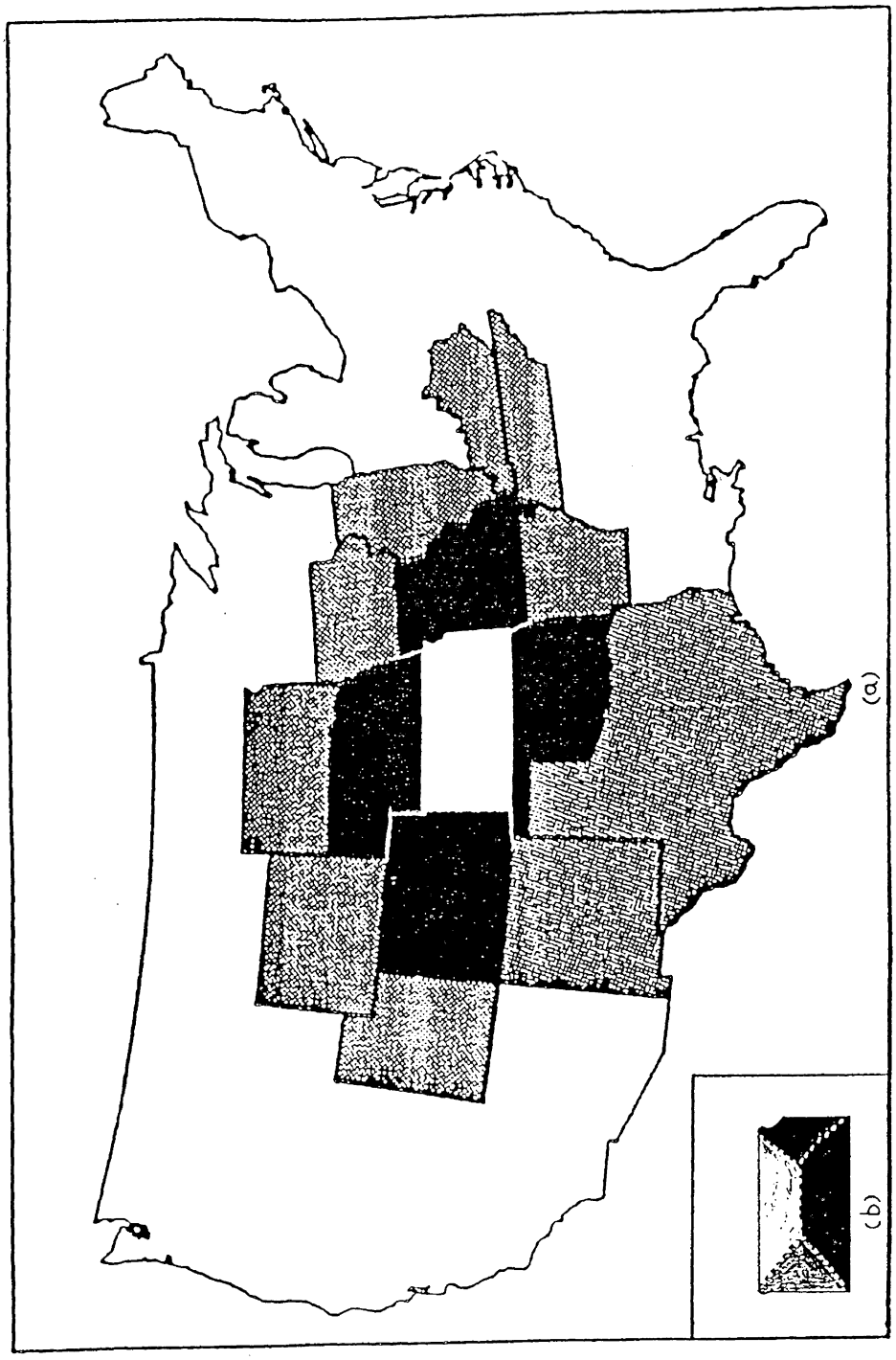


Fig. 1. (a) First- and second-order neighbours to the state of Kansas (from Tobler and Kennedy [1986]). (b) Regions of Kansas annexed by adjacent states.

exist — no more Kansas. The adjacent states would have to occupy the resulting vacuum — presumably by invasion from the previous border with all the armies marching at the same speed until all available territory was occupied. Then, sneakily, Kansas reappears, and steals all the newly acquired territories of the border states. The relative areas stolen from (or originally lost to) the adjacent states, as shown in Fig. 1b, form the weighting of each state's census value in order to estimate a value for Kansas.

Two conclusions can be drawn from this analogy, both arguably 'intuitively reasonable'. Firstly, if a new region is created, it takes an estimated 'value' calculated as the areas stolen from the pre-existing regions, times their observed values, and divided by the total area stolen. If values are considered as elevation, this is equivalent to overall conservation of volume. Secondly, any point in 'vacant territory' is assigned to the nearest adjacent region, probably using Euclidean distance measure. Thus the boundaries of the newly acquired territories would be a form of the 'skeleton', or medial-axis transform, of the region (Kansas) being evaluated. The sub-regions themselves are fairly stable in area, with minor perturbations of common boundaries, and appear to form a good basis for weighted-averaging.

The Point-Interpolation Problem

Thus 'area-stealing' seems an empirically plausible technique for weighted-averaging of 'missing' regions. The approach may also be used for Voronoi diagrams of point data. How is this done? Let us start by taking an exact analogy with the 'Kansas' model. Each data point has an associated 'nearest Euclidean neighbourhood' — or Voronoi region, or Thiessen polygon — calculated by one of various well-known techniques. As with Kansas, let us remove this point, and region. The adjacent polygons will advance and take over the vacant territory — producing identical results to a Voronoi region calculated without the 'Kansas' point in the first place. If the point is re-introduced, territory is re-captured, and the relative areas of the adjacent regions provide weightings for the neighbouring data points. It must be noted that while this writer developed his methods independently on the basis of Tobler's 'Kansas' problem, Sibson [1982] had suggested a polynomial patch technique based on Voronoi region subdivision and spherical polynomials.

A weighted-average of the neighbouring data point elevations will provide an elevation estimate — but we already have our original value! We may thus proceed in one of several ways, depending upon the problem. This new elevation estimate is a 'smoothed' value, based on the neighbours, much like a simple averaging filter used on a raster data set. The weightings may be applied to the slopes in each of the X and Y directions between the central point and each neighbouring point, in order to provide a weighted-average slope estimate of each data point. Or, for proper interpolation purposes — to estimate elevations at a new location — the procedure is reversed. Starting with our original data points and their triangulation and Voronoi polygons (Fig. 2a), a new dummy data point is inserted at the desired sub-sampling location (Fig. 2b), and the neighbouring regions are identified and their areas calculated. Then the dummy point is removed, and the increase in size of the previously-defined neighbourhood regions calculated (Fig. 2c). The weighted average of the neighbouring point elevations is obtained as before, and this interpolated value preserved for further use. Note that the definition of 'adjacent' or 'neighbouring' polygons or data points is internally consistent; if no area is stolen, there is no common boundary and there is no adjacency. Note also that the insertion of a sampling point and polygon may be made at any desired location to suit the sub-sampling scheme required.

Let us examine in one dimension, as in a cross-section, what the implications of the above weighted-average technique are. Figure 3a shows three data points (D.P.) spaced along the X-axis. The dotted line represents the location of a desired interpolated value. In Fig. 3b, the one-dimensional Voronoi regions have been defined, and the 'elevation' value at each data point associated with each region. The Voronoi boundaries are clearly the mid-points between each data point pair. In Fig. 3c, the Voronoi region of the new point to be interpolated has been defined, and has stolen part of the two adjacent regions. These weights are then applied to the elevation values of the two data points to provide an interpolated value at the desired location.

Figure 4a shows the result of this process for a variety of interpolation locations. The result is clearly to produce linear interpolation between data point pairs. Thus our basic area-stealing process in two dimensions is a direct analogue of linear interpolation in one. This, however, is not a desirable end-product — we do not want slope

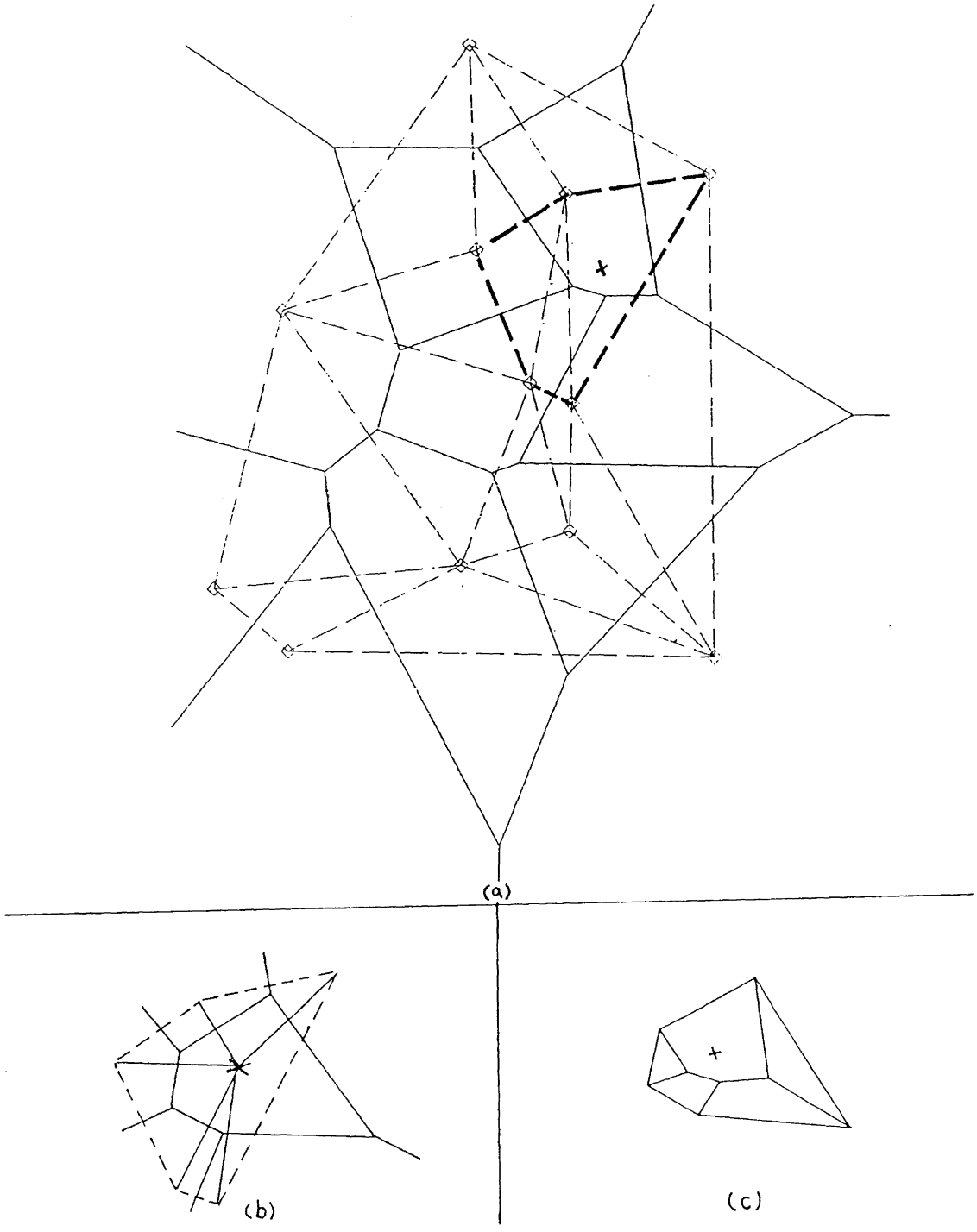


Fig. 2. (a) Original data point distribution with triangulation and Voronoi polygons. (b) The dummy data point is inserted into the data structure, modifying the existing polygons. (c) Upon dummy data point removal, the adjacent polygons annex the vacant area.

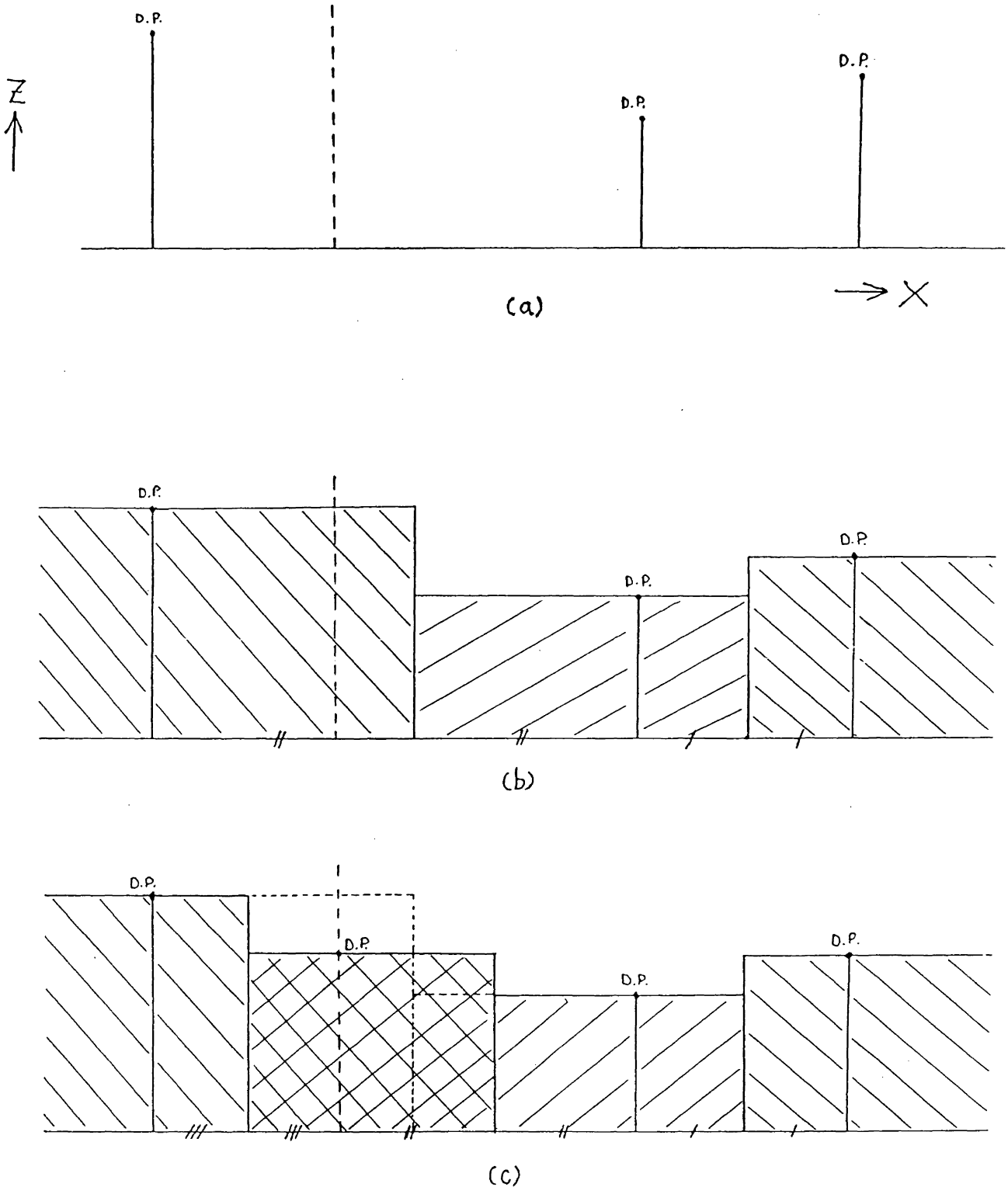


Fig. 3. (a) Original data points. (b) One-dimensional Voronoi of the data points. (c) Interpolation — point Voronoi region introduced.

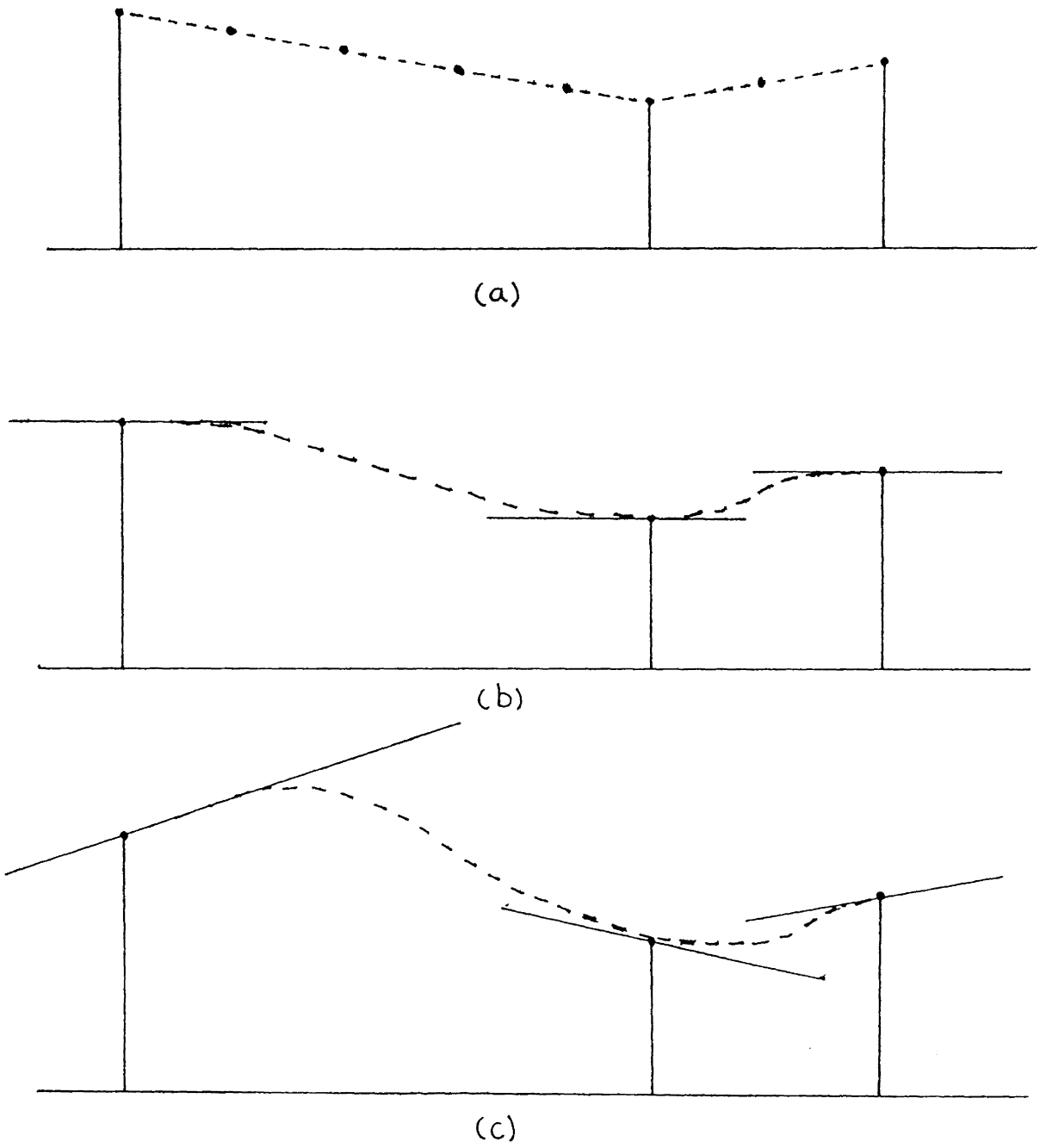
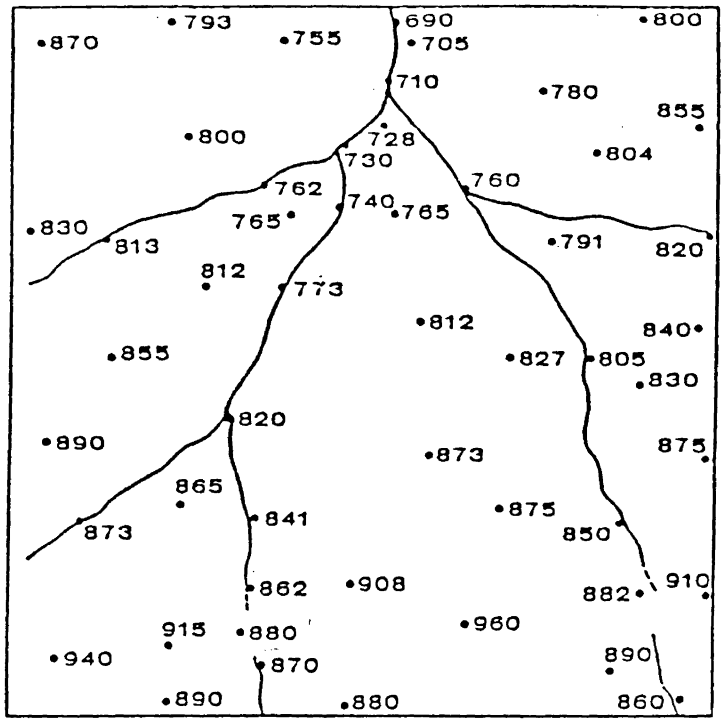


Fig. 4. (a) Result of interpolation using method of Fig. 3c.
(b) Introduction of smooth interpolating function.
(c) Introduction of slopes at data points.

discontinuities at our data points. Figure 4b shows one possible solution. A Hermitian interpolating function is applied to each of the weights prior to summation, so that the first derivative (slope) is zero as the interpolating point approaches a data point. Thus 'smooth' curves (in one dimension) or surfaces (in two) may be produced. Lastly, in Fig. 4c the function at each data point is no longer a simple elevation with no slope but is any general function — in this case planar. In conjunction with the previous Hermitian interpolant, 'smooth' surfaces in the first derivative can be made to match elevation and slope values for any data point distribution.

As discussed in Gold [1984], the selection of a sub-sampling scheme is a basic part of the design of a 'contouring' or interpolation package. Fixed grids are, in general, to be avoided as they impose a fixed sampling frequency that may well be inappropriate to the distribution of the particular data set. Triangulating the data set as given (using the Delaunay triangulation, the dual representation of the Voronoi polygons described above), followed by sub-sampling on a regular n by n grid of sub-triangles within each main triangle, provides an adaptive scheme that compensates for data anistrophy and yet ensures that the interpolated surface still passes through each original data point. Nevertheless, for a variety of applications a grid is the desired output; subsequent perspective views are included to illustrate this point. Figure 5 shows a test data set quoted in Davis [1973] and its triangulation. Figure 6 shows a perspective view of the basic triangulated model. In Fig. 7, the above-mentioned sub-sampling scheme is used providing interpolation between data points but honouring data point values. In Fig. 8, the model has been re-sampled on a grid for display purposes, to facilitate the illusion of depth, but in consequence original data point values are not preserved. Figures 6, 7, and 8 are from Gold [1987].



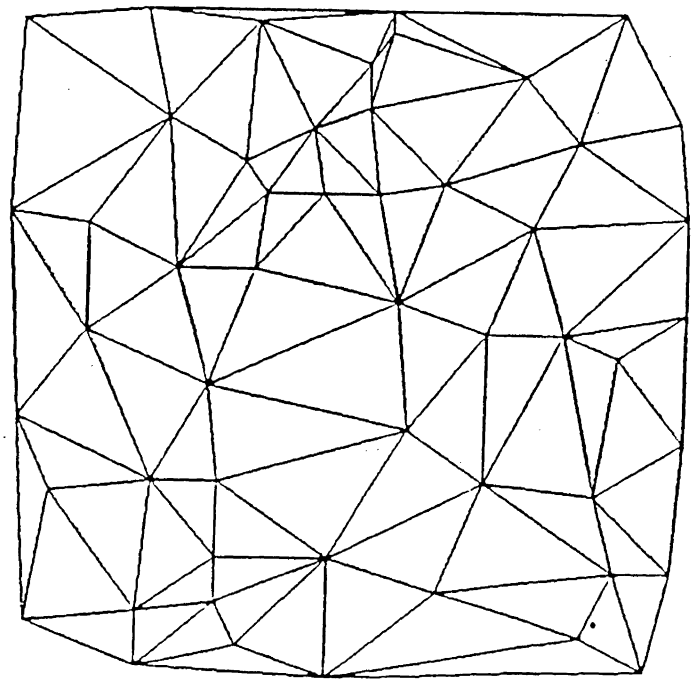


Fig. 5. Davis's data set and triangulation.

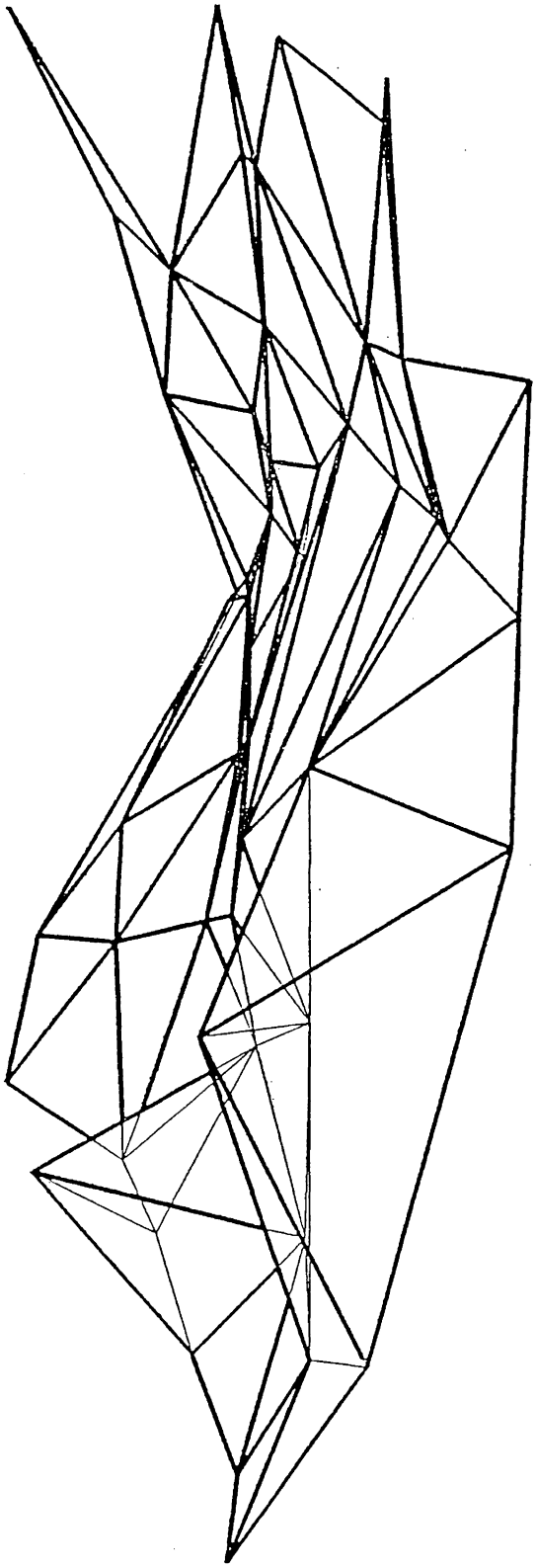


Fig. 6. Perspective view of triangulation from Fig. 5.

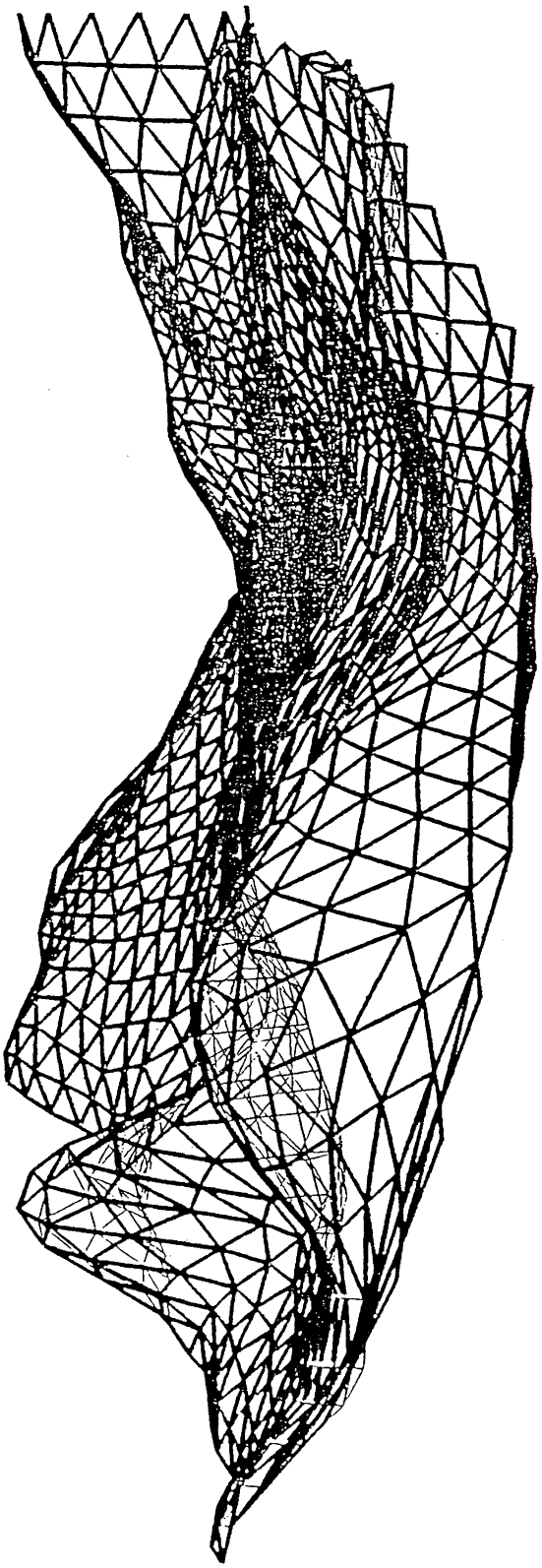


Fig. 7. Interpolation by sub-sampling within triangles.

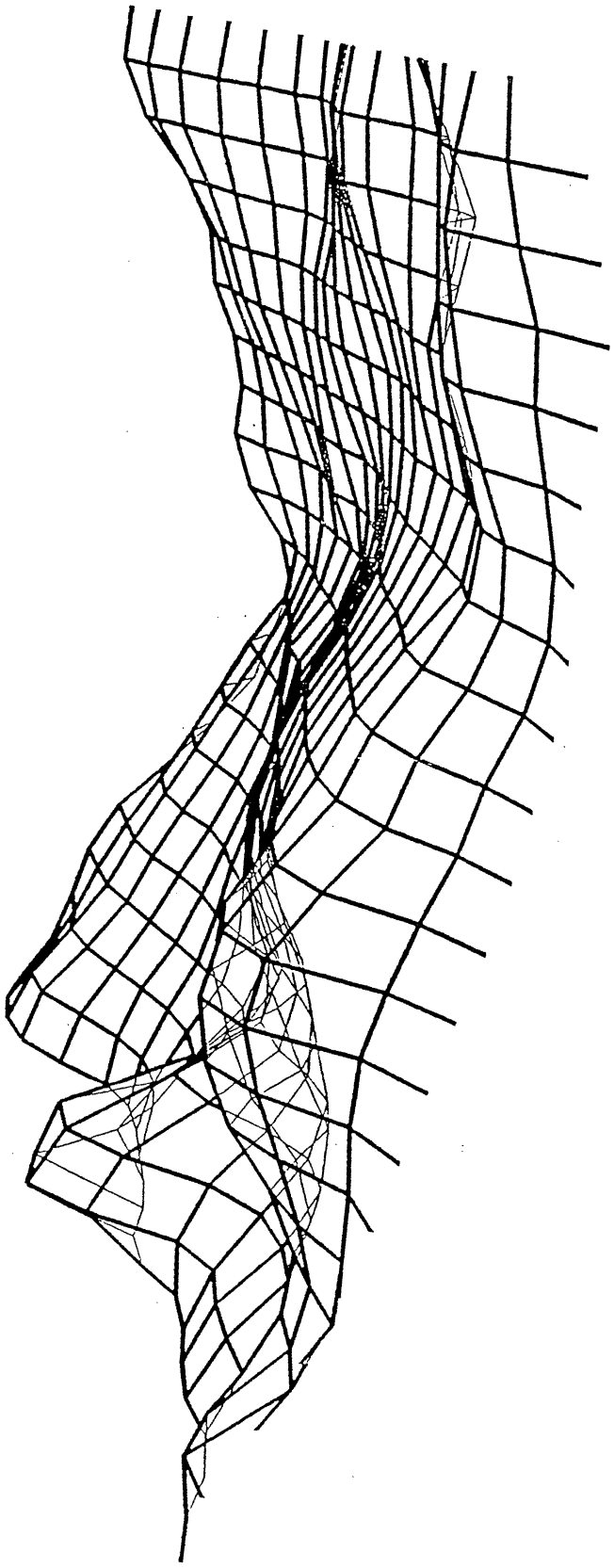


Fig. 8. Interpolation by sub-sampling on a grid.

Conclusions

A variety of issues that are handled only with difficulty, if at all by traditional methods, may be readily resolved by creative theft! The concept of deleting areas, permitting encroachment from adjacent regions, and thus determining the relative contribution of each of these neighbours appears to be sufficiently general to be appropriate for missing-census-district type problems and point interpolation. Point data are first converted to area data by using a Voronoi polygon generator prior to calculation of neighbourhood weights by area stealing. Various functions may be applied to these derived weights in order to provide the desired level of continuity of the interpolant at the data points themselves.

This approach is attractive by comparison with traditional methods both because the input data is honoured under all circumstances and because of the generality of the 'area-stealing' concept.

This research was partially funded by the Energy, Mines and Resources Canada Research Agreement Program.

References

Davis, J.C. (1973). *Statistics and Data Analysis in Geology.* Wiley, New York, 313 pp.

Gold, C.M. (1984). Common sense automated contouring — Some generalizations. *Cartographica*, Vol. 21, No. 2, pp. 121-129.

Gold, C.M. (1987). Ordering of Voronoi networks and their use of terrain database management. *Proceedings, AUTO-CARTO 8*, Baltimore, March, pp. 185-194.

Sibson, R. (1982). A brief description of natural neighbour interpolation. In *Interpolating Multivariate Data*, ed. V. Barnett, Wiley, London, pp. 21-36.

Tobler, W.R. and S. Kennedy (1986). Smooth multidimensional interpolation. *Geographical Analysis*, Vol. 17, No. 3, pp. 251-257.

Discussion

Question: The Voronoi regions, can they be produced for 3-D objects as well as volumes?

Answer: Yes, and of course the nice thing about this is that it is a local process. As Y.C. was talking about with splines, you can guarantee that any point is not going to perturb the whole network. If you use any other definition of a 'good' triangulation, I don't think you can guarantee that it's not going to make waves over in the far corner of the map when we add a new point. That's one of the nice things about the Voronoi process.

Question: The question Max asked was different. Is there an equivalent to Voronoi diagrams in 3-D space?

Answer: Yes, it's obviously not easy, but conceptually, yes. Crystals grow in 3-D space.

Question: Can I come to the blackboard? You have this profile, and you have a point, and you get a Voronoi region around it. Then you have another point, and then you create a Voronoi region around it. Then what you get with your weights is the stolen area from the two sides. Now what if I do an interpolation; I assign that weight back here. Let's say this is the total area of A, and that would be A1, A2, etc. If I now do an interpolation I can now create some kind of a smooth curve between them. I wonder if there is some way like that to interpolate Voronoi diagrams.

Answer: I would expect so if you can define it.

Question: The problem of interpolating the Voronoi diagram is that given a point, what can I know about the relative influence of the two sides? The inverse problem is now we get a smooth curve, and then ask the question: Given that point what would be its Voronoi extent?

Answer: I have not thought about that. But it's interesting because here I am approaching this from the idea of weighted averages. There has been no polynomial function in any of this. I did work with polynomials and I found that I was producing artifacts at triangle boundaries, or patch boundaries, and I was clearly getting first or second derivative discontinuities that showed up on my interpretation of my contour map. One thing about the weighted average process is that the Voronoi is generalizable in principle from one to n dimensions, and the weighted average does not intrinsically have any discontinuities in any derivatives. So this has quite a lot of potential for general process. I believe there is some reasonable physical analogue. It was not accidental that I moved on to talk about line Voronois as well as point Voronois. Take the very interesting question of interpolating from contour intervals. Contour input is not a set of points. If you use points for contour input—bluntly, you get strange things happening. The key with this is to define what you mean by an object. Just as the key to the skeleton coding process is what you mean by an object. You have to count the edges as different objects if you want to look at the boundaries between them. If you use the point Voronoi for this, you have taken all of these different points and called them different. You have boundaries between them. We have a great difficulty making any interpolation here that's different from the contour line itself. Certainly you can use

the slope data, but I have not been very successful. I believe the problem is that we should be treating this as a segment process and not a point process. This is not something that I have worked on in great detail. I know that I have had problems in the point process.

Question: I think that one of the problems is that this is not very attractive because what you produce are not very nice curves. You get paraboloids.

Answer: That is for the Voronoi diagram itself, yes. But that does not affect the smoothness of your interpolation process. That is simply the boundaries of the areas that you steal from.

Question: But you have to calculate areas under them; you have to calculate sections of them.

Answer: Yes, they are not cheap.

Remark: There are probably a couple of strange conditions.

Answer: Well, I have been working on this for a few years. On point processes I have had no difficulty.

Question: I very much like this approach, but I would like to ask you probably a dumb question. Your interpolation is based on the Delaunay triangulation which has resulted from this particular technique. This means that every time you interpolate you use at most three values, or at least the immediate neighbours of the point.

Answer: You use typically a half dozen values, because typically you are going to get six adjacent points when you steal a piece of the region by adding a new point, even if it's an imaginary point.

Question: But it is going to be the immediate neighbours, which means that the number depends on how many immediate neighbours exist.

Answer: That's right. It's entirely a function of what's really there and not how many ought to be there.

Question: So this is really quite good for spatial type applications. I am considering more geological type applications, where you have more points around an area. You remember from the Voronoi diagram that you have an area of influence that has been defined from the sample data. If your area of influence is larger than your immediate neighbours, probably you would have to consider more points than the immediate neighbours and the accuracy of these estimations as well, because in digital terrain models all of our points are supposed to have the same accuracy. If you have points which are farther away but which are more accurate than the immediate neighbours, probably you would like to consider them as well. I am wondering how different would the Kriging method be from your method.

Answer: I think there is a lot of commonality. I think we are talking about a great many of the same underlying processes. There are three points I would like to make. Firstly, Kriging is a weighted average process despite all of the other fun and games that

go into producing it. I haven't mentioned weighted Voronoi diagrams. Supposing you have a radio tower from the Voice of America, which is the most powerful one I can think of now, next door to Annapolis Valley Radio. They are at the same frequency. The zone at which the signal is stronger is going to be a very small region around Annapolis Valley Radio, but even on the other side of it there is going to be a region of the Voice of America. So, you no longer have simple straight line boundaries. You can see easily how the regions enclose other regions. So there are lots of good things you can do with weighted Voronoi. I believe that principle will work. But again, the functional algorithm is going to be relatively difficult. The second point is that we use immediate neighbours for the function at each data point. If the function that we are weighting is somewhat complex, let us say elevation plus slope, then in calculating that elevation plus slope, I use the point's neighbours to produce a weighted average that generates the slope. Then I weight that function, plus others, to interpolate to the new point. So, I have gone to my second generation of neighbours in order to estimate my slope. After all, slope is saying something about the relationship of the point to its neighbours surrounding it. The third point I would like to make is that because Kriging does some very nice things while being able to make some error estimates, we can do the same thing here because we now have no difference between an interpolation point and a real point. The only difference is that we happen to know the value at the real point. Let's hide it and pretend we don't. What value do we get? We have weighted average estimates. We can start talking about the normal distribution of estimates, and then start saying how does this distribution relate to — ah ha!. I fooled you. I'll bring back my real data point. So we have quite a lot of comparison processes available.

Question: Also one important thing would be to try to model discontinuities in your structure?

Answer: That's relatively easy because with any weighted average process you simply have to have a structure which allows you to identify the points which count and the points which don't count.

Question: But with all interpolation methods it is very close to black magic. You have a limited amount of information and you produce new data. You put in some understanding of what is likely to be there. For example, Y.C. in his talk pointed out very clearly that you assume that the surface of a curve is continuous in some sense. So you make some additional assumptions. I think in most interpolation methods I have seen there is other information. But very seldom is it explained exactly what the additional knowledge is that is put in to produce that new information. You have to put something in.

Answer: It's a cautionary tale to realize that despite all these fancy pictures at the bottom, the top is data.

Question: Only the points at the top are data. That's the only thing you really know. Everything else is based on assumptions, which are perhaps justified and perhaps not.

Answer: That's true. One of the things that has annoyed me the most with many contouring packages is that they give you an accuracy figure. You get 98.5% and you say that sounds good, but what does that mean? That means that they have approximated the data to match the data points 98.5% of the time. That's all they are

saying. So they should have done it to 100%. It says nothing about the function that's underlying the process.

Question: You cannot say anything useful unless you know something about what process affected the surfaces between the data points. What type of surface you expected.

Answer: But, in general, we do know something. We know that we can mislead by using arbitrary processes — by mis-selecting the number of neighbours we wish to use in such a manner that we throw out a point because it has fallen out of an octant or it has moved over a fraction. It's just fallen out of our circle or radius process or whatever, and it still has a significant weight in our function. So we have a strong interaction between points and weight, and one of the strengths of the Voronoi process is that weights and neighbours are directly related.

Question: If you use one of the methods you used at the beginning and you shift the grid a little bit ,you get a different result.

Answer: If you are using a grid, yes. You won't with mine.

Remark Yours is invariant. That's one thing I like about it. We cannot say that one of the results by one of the non-invariant methods is correct and the other is wrong. We just can't say that.

A SURVEY OF THE CONVEX HULL PROBLEM

J.D. Horton
School of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3
JDH@UNBMVS1.bitnet

1. Introduction

The convex hull problem is one of the most studied problems in the new area of computational geometry. Given a finite set S of points in d-dimensional real space, how can one find the **convex hull** of S, $CH(S)$ = the smallest convex set that contains S? For a finite set S in two-dimensional space $CH(S)$ is a finite polygon, except in some degenerate cases in which case $CH(S)$ may be a line segment, a single point, or when S is empty, is the empty set itself. In three dimensions, $CH(S)$ is a finite convex polyhedron. In higher dimensions, when S is not embeddable in a lower-dimensional affine subspace, $CH(S)$ is a finite convex polytope.

A solution to the convex hull problem requires that $CH(S)$ be described in some manner. Usually only the boundary of $CH(S)$ is described. In two dimensions a list of the vertices in order around the polygon is sufficient. In higher dimensions, a list of all faces of the polytope can be given as the solution to the problem. However, because the number of facets of a polytope can be much larger than the number of vertices, sometimes it may be sufficient to list simply the vertices of the polytope, or even just give a fast algorithm to decide whether a point is in the convex hull or not.

A set S is said to be **convex** if for any two points of the set a and b, any convex combination of a and b, $\alpha a + \beta b$, where α and β are positive real numbers and $\alpha+\beta = 1$, is also in the set S. A convex **polytope** in d-dimensional real space, R^d, is the intersection of a finite number of closed **half-spaces**, that is, all points of R^d lying on one side of a hyperplane. A hyperplane is said to **support** a convex polytope P if P is

contained in one of the half-spaces determined by the hyperplane and the hyperplane intersects P. In this paper we deal with only finite or bounded polytopes. The convex hull of a finite set of points is always a finite convex polytope; conversely, a finite convex polytope is the convex hull of a finite set of points.

A **face** of a convex polytope P is the intersection of (the boundary of) P with a hyperplane that bounds P. A face is always a convex polytope itself. A face is said to be k-dimensional, or k-face, if the smallest flat (translation of a vector subspace) in which it is contained is of dimension k. If the convex polytope is in d-dimensional space, then the (d-1) faces are called **facets**, and the (d-2) faces are called **subfacets. Vertices** are 0-faces and **edges** are 1-faces. In the plane, facets are edges and subfacets are vertices.

Much of the material in the paper is contained in Chapter 3 of the book *Computational Geometry, an Introduction* by Preparata and Shamos [1985], where more detailed descriptions of the algorithms can be found. The book is highly recommended as an introduction into the rapidly expanding area of computational geometry. A few more recent papers are also cited, principally concerning implementations of algorithms.

For all the algorithms given in this paper, it is assumed all points are in general position. That is, no two points are the same, no three points are collinear, no four points are co-planar except in the two-dimensional case. In general, no k+2 points are in a k-dimensional flat if $k<d$. Some of the algorithms may not be affected by this assumption, but others can be more complicated to implement and/or analyse. However, the concept of the algorithms are not affected by such a simplification.

Order notation used in this paper: $f(n) = 0(g(n))$ means that there is a real constant $c>0$ such that for all $n, f(n)<cg(n)$. $f(n)=\Omega(g(n))$ means that there is a real constant $c>0$ such that for all $n\, f(n)>cg(n)$. $f(n)=\theta(g(n))$ means $f(n)=0(g(n))$ and $f(n)=\Omega(g(n))$. Thus there exist constants c and C such that $0<c<C$, and for all n, $cg(n)<f(n)<Cg(n)$.

The time-**complexity** of an algorithm is defined as maximum number of operations that the algorithm may be required to perform on an input of a given size, i.e., it is a

function of the length of input. The input of a convex hull algorithm is a set of n points in d-space, which is proportional to dn. However, since the effect of n and of d may be quite different on the number of operations required, and d is much smaller than n, the complexity of a convex hull algorithm is often expressed as a function of n only. The dimension d is considered to be a constant. On the other hand, the output of some convex hull algorithms is often as important as the input in determining the number of operations performed. Thus, sometimes the complexity is expressed in terms of n (the original number of points), h (the number of vertices of the convex hull), and/or f (the number of facets in the convex hull).

Complexity is usually given using the order notation above. The time that an actual implementation takes is proportional to the number of operations that it requires, and the order notation used is also proportional to the actual number of operations. Thus unless the constants involved are large or the time to perform an operation is short, the order notation gives a good idea of how effective an algorithm is.

For example, suppose one has a computer that can perform 10^6 or 10^7 operations per second, and that the constants of proportionality used in the order notations are between 10 and 100. Then an algorithm of complexity $0(n \log n)$ could handle any set of 1000 points in under a second. A complexity of $\theta(n^2)$ means the algorithm could handle any set of a few hundred points in less than a minute, but some set of 1000 points would take at least 10 seconds, if not longer. If the average complexity was $\Omega(n^3)$, then a set of 200 points would very likely take longer than a minute, maybe more than an hour.

2. In Two Dimensions

The two-dimensional case has certainly been the most studied. Many algorithms are known that solve the convex hull problem in 'optimal' time, i.e., within a constant factor of the minimum number of operations required. However, the best algorithm to use is unclear, and depends upon the application. Below are listed several techniques.

Before proceeding to algorithms that solve the problem, note that an algorithm that solves the convex hull problem can be used to sort È numbers.

Let $x_1, x_2, x_3, ..., x_n$ be distinct positive numbers. Construct the n points $P_i = (x_i, x_i^2)$ on the parabola $y=x^2$. A convex hull algorithm applied to these points returns the points P_i in order around the polygon, since all these points are vertices of the convex hull. The point furthest to the left can be found in linear time. Following counter-clockwise around the polygon determines the order of the x_i. Thus any convex hull algorithm cannot take less than the complexity of the sorting problem minus $0(n)$ Since sorting is $\theta(n \log n)$ under the common models of computation, the convex hull problem has a lower bound worst case complexity of $\Omega(n \log n)$. This lower bound has been extended to the problem of just identifying the vertices, without putting them into order, under the fixed order algebraic decision tree model of computation [Preparata and Shamos, 1985, pages 96-97].

2.1 The Graham Scan

Graham [1972] published the first algorithm that is less than quadratic in the number of points input. It consists of four steps (Figure 1):

1. Choose a point q inside the polygon.

2. For each point p in S, calculate the angle formed between $\overline{q}\,\overline{p}$ and the x-axis.

3. Sort the points of S by angles obtained in step 2, to obtain a polygon (p_0, $p_1, p_2, ..., p_n=p_0$).

4. Walk around the polygon removing all reflex vertices (p_i is a **reflex** vertex if the angle determined by $p_{i-1}\, p_i\, p_{i+1}$ is greater than π). The remaining vertices determine the convex hull of S.

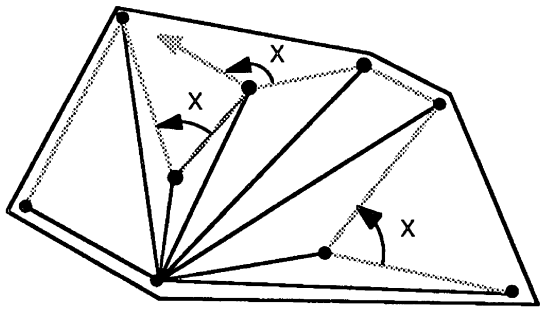


Figure 1. The Graham scan.

A point inside the polygon can be found by choosing the centroid of any three non-collinear points of S. An alternative method is to let q be an extreme point of S, for example, the point of minimum y-value, and insert q into the appropriate position in the polygon. In either event, this step can be calculated in $0(n)$ time, where n is the number of points in S.

In the second step, the angles themselves need not be calculated, but only the slopes and the quadrants. This removes the need to calculate the inverse tangent function. This step is also $0(n)$.

The third step is $\theta(n \log n)$ for most common models of computing.

The last step is $\theta(n)$, since every angle has to be checked, and at most n-2 vertices need to be removed. One cautionary note: when a vertex is removed, its neighboring vertices must be checked for reflexivity, even if these vertices have been checked previously. Thus the overall algorithm is $\theta(n \log n)$, which is optimal as shown by the lower bound derived from sorting.

2.2 The Gift-Wrapping Algorithm

Chand and Kapur [1970] proposed an algorithm to solve the convex hull problem in d dimensions. A two-dimensional algorithm similar to Chand and Kapur's was developed independently by Jarvis [1973], which has since been called Jarvis's march (Figure 2). An outline of the algorithm:

1. Find the point of S of minimal y-value, P_0.

2. Find the point p_1 in S such that the slope of $\overline{P}_0 \overline{P}_1$ is minimum.

3. For each i, i=1,,2,..., n-1, find the point p_{i+1} in S-{$(p_1, p_2, ..., p_i\}$ such that the angle $p_{i-1}\, p_i\, p_{i+1}$ is maximum (alternatively, the direction of $\overline{p}_i\, \overline{p}_{i+1}$ is as close as possible to the direction of $\overline{p}_{i-1}\, \overline{p}_i$. Stop when p_0=p_n. Then $p_1, p_2, p_3, ..., p_n = p_0$ is the (sequence of vertices in the order on boundary of the) convex hull of S.

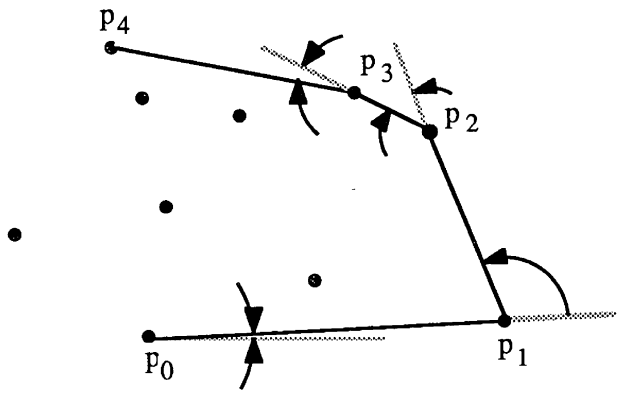


Figure 2. The Jarvis march.

Steps (1) and (2) are clearly linear ($\theta(n)$) as is the interior of the loop in step (3). Since the loop is performed h times, where h is the number of vertices on the boundary of the convex hull, Jarvis's march has time-complexity $0(nh)$. Although in the worst case this is $0(n^2)$, if h is known to be small the algorithm is fast and easy to program, and can be competitive with the 'optimal' Graham's scan. Graham's scan is only known to be optimal in the worst case with all the points being vertices of the convex hull. In fact, Jarvis's march has a better expected asymptotic complexity for points normally distributed in the plane, since for n such points, the expected value of h is on the order of the square root of log n [Raynaud, 1970]. This leads to the question, can we do better in other cases?

2.3 The Ultimate Planar Convex Hull Algorithm

Kirkpatrick and Seidel [1986] present a planar convex hull algorithm that is $0(n \log h)$. They have also shown that this is optimal, i.e., that any algorithm will take $\Omega(n \log h)$ on some set of n points with h vertices on the hull. When $h=n$, this reduces to $\theta(n \log n)$, but it is rare for the number of vertices on the convex hull to be linear in the number of points in the set, as can be seen from the following set of theorems extracted from Preparata and Shamos [1985, p. 145]. In all cases, n points are chosen independently at random from some distribution in k-dimensional space, $E(h)$ is the expected number of vertices of the convex hull of those n points, and $E(f)$ is the expected number of facets of the convex hull.

Table 1

Distribution	$E(h)$	$E(f)$
Uniformly from a plane convex region	$\left(\frac{2r}{3}\right)(r + \log n) + 0(1)$ [Renyi and Sulanke, 1963]	$E(h)$
Uniformly from a k-dimensional sphere		$0(n^{(k-1)/(k+1)})$ [Raynaud, 1970]
Thus uniformly from circle: uniformly from sphere:	$0(n^{1/3})$ $0(n^{1/2})$	$0(n^{1/3})$ $0(n^{1/2})$
Normally from k-dimensions	$0((\log n)^{(k-1)/2})$ [Raynaud, 1970]	—
All components independently chosen from **any** continuous distribution (e.g., hypercube)	$0((\log n)^{(k-1)})$ [Bentley et al., 1978]	—

It is remarkable that the number of vertices on the hull is considerably fewer than the number of points in the set in all cases. One might expect therefore that this result would still be true in an applied situation, and I would not disagree unless some obvious reason to the contrary could be given, such as experimental evidence, or the distribution of points were confined in a non-convex region like a ring. On the other hand, note that these are only asymptotic results, except for the polygon case, and that the constant of proportionality may be large, as Stewart's [1987] results in the uniform k-dimensional hypersphere case indicate. (In 10 dimensions, for a set of 1000 random points uniformly distributed in a hypersphere, only 23 were not vertices of the convex hull.) Thus this algorithm seems to be considerably better than the other algorithms mentioned previously.

The algorithm starts by finding the two points with extreme x-values, p_{left} and p_{right}, which are on the hull. Then the upper and lower hulls joining these points are found separately. We describe the method to find only the upper hull as the lower hull can be found separately and in the same way. The algorithm is a divide-and-conquer type that uses a marriage-before-conquest procedure instead of the more common conquest-before-marriage procedure. One usually thinks of divide-and-conquer as consisting of:

1. Subdivide problem P into problems P_1 and P_2.
2. Solve problems P_1 and P_2 separately (conquer).
3. Combine the solutions to problems P_1 and P_2 to get a solution for P (marriage).

The 'ultimate' algorithm reverses the last two steps. Given the set S of points above the line P_{left} and P_{right}, the algorithm proceeds:

1. Choose a vertical line $x=a$. Let S_1 be the points of S to the left of the line, and let S_2 be the points to the right of the line.
2. Find the **bridge** between S_1 and S_2, that is, find the line segment $\overline{p}_1\,\overline{p}_2$ that joins a point of S_1 to a point of S_2 and is on the convex hull (marriage).
3. Find the upper hull joining P_{left} to p_1 from set S_1; and the upper hull joining p_2 to p_{right} from set S_2 (conquest).

The ordinary conquest-before-marriage algorithm can also be used, but it apparently leads to a worst case analysis of $\theta(n \log n)$. The advantage of the marriage-before-conquest algorithm is that no time is wasted calculating the convex hull of sets of points that later are found to be in the interior of the whole set.

Both steps 1 and 2 have some possible variations in implementation. In step 1, the value of a must be chosen. The original algorithm chooses a to be the median x-value of the set S, so that S_1 and S_2 are evenly divided. The calculation of the median is asymptotically a linear-time algorithm [Bentley et al., 1978], but it has a relatively large constant. Kirkpatrick and Seidel therefore recommend choosing the x-value of a random point of the set to be a, which leaves an expected $0(n \log h)$ algorithm with a much smaller constant, but is $\Omega(nh)$ in the worst case.

The calculation of the bridge in step 2 can be found using the general linear-time algorithm for linear programming by Dyer [1984]. Kirkpatrick and Seidel actually define in detail the method finding the bridge.

Algorithm to find the bridge between S_1 and S_2 across the vertical line $x=a$:

1. Pair off the points in $S_1 \cup S_2$.
2. Choose a slope s.
3. Find the point p of $S_1 \cup S_2$ through which a supporting line for the convex hall passes.
4. If p is in S_1 (p in S_2 is analogous) then for any of the pairs of points chosen in step (1), say p_1 and p_2, if the slope of the line through p_1 and p_2 is greater than s, then the left point of p_1 and p_2 cannot be an endpoint of the bridge and should be removed from S_1 or S_2.
5. Steps (1) through (4) are repeated until S_1 and S_2 both have one point each.

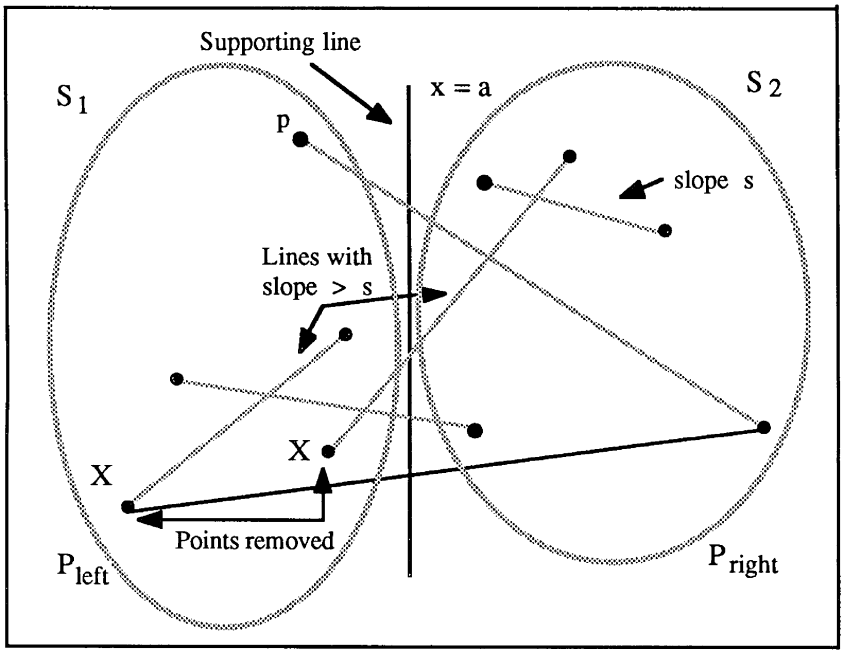


Figure 3. Removing points while searching for the bridge.

The algorithm can be guaranteed to be linear by choosing s to be the median of the set of slopes of the chosen pairs of lines. Then one-fourth of the points in $S_1 \cup S_2$ are removed at each step. Thus the complexity of each step will be about three-fourths of the complexity of the previous step through the loop. The total complexity then will be about four times the complexity of the first pass through the loop, which is linear.

However, again the median of a set of numbers, in this case the slopes, is calculated. Thus Kirkpatrick and Seidel again recommend choosing a slope at random from the pairs of points. Again the expected complexity decreases by a constant factor, while allowing a possible quadratic worst case. In fact, with the two choices being made at random, the worst case becomes cubic, $\Omega(n^3)$ although such a chance is extremely remote.

Unfortunately, the above algorithm is not so good in practice. McQueen and Toussaint [1985] have experimented with the algorithm, with improvements added, but found that other algorithms are better for fewer than 1000 points.

2.4 Quickhull

Another simpler algorithm has been recommended by several people independently (Eddy [1977], Bykat [1970], Green and Silverman [1979], Floyd [1976] as reported in Preparata and Shamos [1985, p. 106]). Apparently it was inspired by Hoare's [1962] Quicksort algorithm. The algorithm starts like the ultimate algorithm by finding the left-most and right-most point p_{left} and p_{right}, and then finding the upper and lower hulls separately. The algorithm is also recursive, but the idea is simpler. At any given stage, the point furthest from the line segment under consideration is found. This can easily be done in linear time, and the point found is also on the hull. Then two new line segments are formed, over which the algorithm is performed again. Algorithm to find the upper convex hull of a set S over a line segment $\overline{p}_{\text{left}}\,\overline{p}_{\text{right}}$ (Figure 4.):

1. Find the point p furthest from the line $\overline{p}_{\text{left}}\,\overline{p}_{\text{right}}$.
2. Partition the points in S into L, R, and points to be thrown away inside the triangle $p\,p_{\text{left}}\,p_{\text{right}}$. L consists of the points above line $\overline{p}_{\text{left}}$ p and R consists of the points above $p\,\overline{p}_{\text{right}}$.

3. Find the upper convex hull of L over $\overline{p}_{left}$, p, and the upper convex hull of R over $p\ \overline{p}_{right}$: (using recursion).

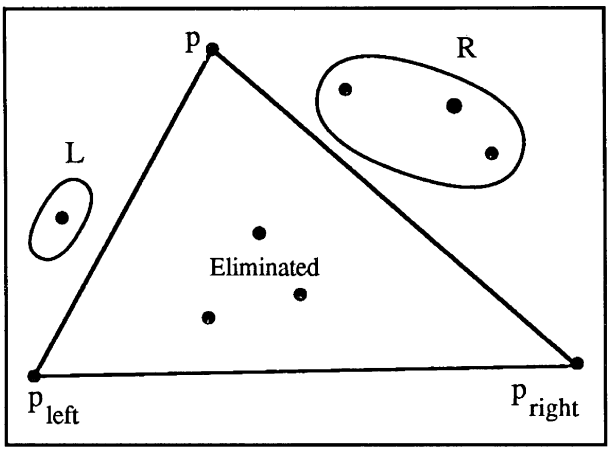


Figure 4. A step in Quickhull.

Each step in the algorithm is clearly linear, and with a small constant. Also, at each step many points will be eliminated. In fact, if the points are chosen from some uniform distribution, the area of the triangle will exceed the areas left for L and R, and hence one would expect more than half the points to be eliminated at each step. This leads to expected linear behavior, with a small constant. With points distributed even more to the centre, a larger fraction of points can be expected to be eliminated and so the algorithm should run to completion faster still.

However, a poor set of points can lead to quadratic behaviour for Quickhull. For example, if the points were all on the convex hull, say on the downward-facing parabola $y = -x^2$ (Figure 5), and the points were (0, 0), (1, -1), (1/2, -1/4), (1/4, -1/16), (1/8, -1/64), ..., dividing the x-value by two for each point, then the points would all be found in the order given above, and all points not yet recognized as being on the convex hull, would have to be searched to find the next point. R would remain empty for the whole algorithm.

Despite this possible bad case, I would recommend Quickhull for most practical situations. It is expected to be very fast for many distributions. It may run slowly if the points tend to be scattered near the hull in irregular clusters.

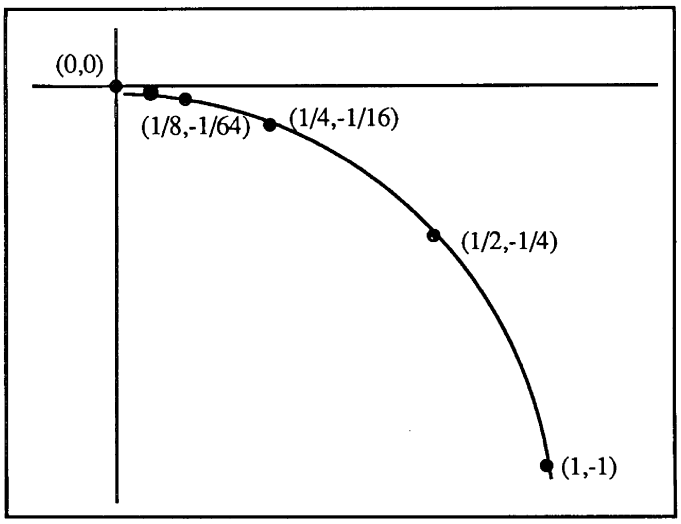


Figure 5. A bad case for Quickhull.

2.5 Preprocessing

Devroye and Toussaint [1980] have pointed out that for most practical purposes, the algorithm chosen may not be very important, as long as some preprocessing is performed. They suggest finding the points furthest in eight directions: max x, min x, max y, min y, max $x+y$, min $x+y$, max $x-y$, max $y-x$. The eight (or fewer) points are then formed into a polygon and all points inside are removed. It can be seen that for many natural distributions the number of points left are likely few relative to the original number, and that any $0(n \log n)$ algorithm can process them much faster than the original set. In all the cases mentioned in Table 1, except for uniform within a circle, the number of points left can be expected to be less than linear. Then the remainder of the algorithm takes less than linear time in the original input — the linear time preprocessing does most of the work. They also point out that even Graham's original algorithm can be implemented in linear time using a bucket (or radix) sort, under mild restrictions on the input.

3. The Three-Dimensional Case

In higher-dimensional cases, the known algorithms are not as plentiful nor as fast as the two-dimensional case. In fact, only three dimensions admit a subquadratic algorithm. One algorithm, due to Preparata and Hong [1977], is a variant of divide-and-conquer (Figure 6).

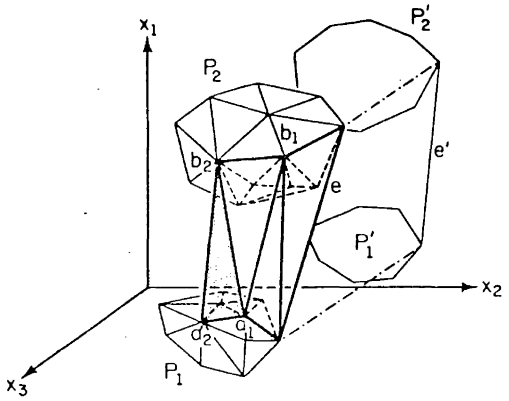


Figure 6. Divide and conquer in three dimension
(from Preparata and Shamos [1985, p. 137]).

The first step is to sort the points by z-value. The set is then partitioned into two equal parts S_1 and S_2, with S_1 having the lower z-values, and S_2 having the higher z-values. The convex hull of S_1 and of S_2 are then found recursively. Finally the two convex hulls are merged. (Note this is a standard conquest-before-marriage algorithm.) If the merge operation can be done in linear time, then the overall algorithm is $0(n \log n)$. Hence we must show that the merge operation is linear.

The merge operation is like a Jarvis's march but in three dimensions. More commonly, the term 'gift-wrapping' is used for this process. First note that $CH(S_1)$ and $CH(S_2)$ are separated by a plane $z=a$. A set of faces and edges must be found

joining $CH(S_1)$ and $CH(S_2)$. The intersection of the new faces with $CH(S_i)$ forms a cycle of vertices and edges around $CH(S_i)$. Points, lines, and faces 'inside' this cycle must be removed, while points, lines, and faces 'outside' will become part of $CH(S_1$ $S_2)$. Since the number of edges and faces of a three-dimensional polyhedron is linear in the number of vertices, these changes can be checked in linear time. Thus we only need to find the faces that form the bridge between the two polyhedra in linear time.

Assume we have a face F in the bridge, with the edge v_1v_2 with v_1 in S_1 and v_2 in S_2. The face F defines a plane. This plane can be rotated (wrapped) around the line through v_1 and v_2 until the plane hits a point v of S_1 S_2. This can be done by checking the angle between the plane of F and the plane determined by vv_1v_2. The point v that determines the smallest angle is chosen. Then vv_1v_2 forms a new face of the bridge.

Not any point can be v. In fact, since either vv_1 or vv_2 is an edge of the convex hull, one of these edges must be in the polyhedra $CH(S_1)$ and $CH(S_2)$. Thus only the vertices neighboring v_1 and v_2 need to be searched to determine v.

Once v has been found, the face F can be replaced by the face vv_1v_2. If v is in S_1, then this new face can be wrapped around the edge vv_2 to get another face. Continuing, eventually we return to the face at which we started, which completes the process.

We have yet to mention how to get the first face of the bridge. One method is to project the polyhedra onto the XZ plane, find a supporting line connecting the two polygons that are formed, and take the inverse image of the supporting edge. This gives an edge u_1u_2 of the bridge. Instead of starting with a face, take the plane through u_1u_2 and perpendicular to the xz plane, and wrap it around u_1u_2. This gives a starting face.

We must still show that this procedure can be made to take only linear time. Given a face F and a bridge-edge v_1v_2, the point v can be found from the set of vertices of $CH(S_1)$ attached to v_1, and the set of vertices of $CH(S_2)$ attached to v_2. The vertices of $CH(S_1)$ attached to v_1 can be compared with each other, as can the vertices of $CH(S_2)$ attached to v_2. The vertices should be checked in order around v_i, from the edge in the previous face, through the edges that disappear into the interior of $CH(S)$ (if any), around to the edge that is next in the cycle around $CH(S_i)$, and one edge beyond. Each

comparison of vertices around v_i eliminates an edge of $CH(S_i)$ from being a possible convex hull edge of S, except for the one comparison beyond the best edge. Thus the number of such comparisons is proportional to the number of edges in $CH(S_1)$ which is linear in the number of vertices of S_i. The comparison of the best vertices in each of S_1 and S_2 generates a new edge of the bridge. Again the number of comparisons is linear in the number of vertices. Thus the procedure is linear, and the whole algorithm is $0(n \log n)$.

4. Higher Dimensions

For the d-dimensional problem, $d>3$, the number of facets is more than linear, in fact $0(n^{\lfloor d/2 \rfloor})$ [Klee, 1966]. Thus to simply list the facets is at least quadratic, in fact exponential in d. Several algorithms have been proposed.

4.1 Gift-Wrapping

Chand and Kapur [1970] suggested the giftwrapping algorithm before any of the common planar algorithms had been invented. The idea is a generalized version of Jarvis's march and the Preparata-Hong three-dimensional bridging technique, although it predates both algorithms. The algorithm depends upon the fact that any subfacet (d-2 dimensional face) is a subface of exactly two facets.

Given a facet F, with a subfacet that is not in any other facet that has been found so far, a point is chosen which makes a plane with the subfacet that has the smallest angle with the plane containing facet F. This is called wrapping the plane of facet F around the subfacet. This new point together with the subfacet makes a new facet of the convex hull. A list of facets found and the subfacets which exist in precisely two of these facets must be kept. Bhattacharya [1982] has shown that the complexity of this algorithm is $0(n^{\lfloor d/2 \rfloor + 1})$ in the worst case, a factor n greater than the length of output in the worst case.

4.2 The Online Method

Dijkstra [1976] was perhaps the first person to describe a convex hull algorithm that accepts one point at a time and updates the convex hull. The method accepts one point at a time, and merges this single point with the convex hull of all the points obtained so far, to get a new convex hull. In two dimensions this technique can be performed in $\theta(\log n)$ time for each point, which is optimal [Preparata and Shamos, 1985, p. 117]. In fact in two dimensions the technique can be extended to include removal of points from the set and still keep the convex hull updated. The technique of Overmars and van Leeuwen [1981] can do the update in $\theta(\log^2 n)$ time which may not be optimal.

Kallay's method [1981] requires keeping a list of all faces in the convex hull. As each point arises, each face is checked to see whether it stays in the convex hull, whether the new point can be added to the face to form a new face, or if the face becomes buried inside the convex hull. The algorithm is $0(N^{\lfloor d/2 \rfloor + 1})$, the same as the gift-wrapping algorithm.

Seidel [1981] also describes an algorithm similar to Kallay's, but works with the dual problem (i.e., points become hyperplanes, hyperplanes become points, etc.). He manages to get a running time of $0(N^{\lfloor (d+1)/2 \rfloor})$ which is optimal for even d for the worst case.

Swart [1985] gives a detailed analysis of these algorithms, and comes to the conclusion that his own algorithm, based on Chand and Kapur's giftwrapping algorithm, is better than Seidel's. The main reason is that Seidel's algorithm may have to calculate many facets that will disappear at a later stage. It is easy to give examples that cause many more facets to be created and destroyed than actually are on the convex hull. In fact, Swart shows that Seidel's algorithm can actually take time exponential in the length of input (the number of points) and output (the number of facets).

4.3 Implementations

Not all the reported algorithms have been implemented. For example, Seidel never implemented his algorithm, nor does he know anyone who has. In contrast, Chand and Kapur implemented their algorithm, and used it to solve practical problems with 1000

points in six dimensions. However, they only published timings for up to twenty points (thirteen seconds).

Stewart [1987] implemented an on-line algorithm which can handle 150 points in four and seven dimensions easily. However, it can handle only about 25 points in ten dimensions in reasonable time and space. He also developed a new algorithm that had slightly better practical results. It is similar to Kallay's in that one point is added at a time to the convex hull of all points considered so far. However, the point to be added is chosen to be the point furthest from a known face of the present convex polytope, like in Quickhull. Thus it is not an on-line algorithm. The advantage of this algorithm is that at each step either a new vertex is found or a new facet confirmed as belonging to the final convex hull. As no vertex is ever removed at a later stage, the flaw in Seidel's algorithm (and Kallay's) that Swart found is partially corrected. Thus, the algorithm may be competitive with any known algorithm, including Chand and Kapur's algorithm.

The major algorithm in Stewart's work is much faster. However, it only finds the vertices on the convex hull, and not the facets. Each point is processed one at a time. A search is made that finds either (1) a simplex that encloses the point thus showing that the point is not a vertex, or (2) a direction in which the point is an extreme point of the set, and hence is a vertex. The algorithm can find the convex hull of 1000 points in ten dimensions in about ten seconds. Unfortunately, no upper bound is known for this algorithm. In fact, it is not yet proven that the algorithm always stops. Timing tests indicate the algorithm may have a time-complexity that is better than $0(dn^2)$.

References

Bentley, J.L. H.T. Kung, M. Schkolnick and C.D. Thompson (1978). On the average number of maxima in a set of vectors and applications. *J. ACM 25*, 536-543.

Bhattacharya, B.K. (1982). Worst-case analysis of a convex hull algorithm. Unpublished manuscript, Dept. of Computer Science, Simon Fraser University.

Bykat, A. (1970). Convex hull of a finite set of points in two dimensions. *Info. Proc. Lett. 7*, 296-298.

Chand, D.R. and S.S. Kapur (1970). An algorithm for convex polytopes. *J. ACM 17*, 78-86.

Devroye, L. and G.T. Toussaint (1980). A note on linear expected time algorithms for finding convex hulls. *Computing 26*, 361-366.

Dijkstra, E.W. (1976). *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, N.J.
Dyer, M.E. (1984). Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comp. 13*, 31-45.
Eddy, W. (1977). A new convex hull algorithm for planar sets. *ACM Trans. Math. Software 3*, 398-403.
Graham, R.L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Info. Proc. Lett. 1*, 132-133.
Green, P.J. and B.W. Silverman (1979). Constructing the convex hull of a set of points in the plane. *Computer Journal 22*, 262-266.
Hoare, C.A.R. (1962). Quicksort. *Computer Journal 5*, 10-15.
Jarvis, R.A. (1973). On the identification of the convex hull of a finite set of points in the plane. *Info. Proc. Lett. 2*, 18-21.
Kallay, M. (1981). Convex hull algorithms in higher dimensions. Unpublished manuscript, Dept. of Mathematics, Univ. of Oklahoma, Norman, Oklahoma.
Kirkpatrick, D. and R. Seidel (1986). The ultimate planar convex hull algorithm. *SIAM J. Compu. 15*, 287-299.
Klee, V. (1966). Convex polytopes and linear programming. *Proc. IBM Sci. Comput. Symp. Combinatorial Problems*, IBM, White Plains, N.Y., pp. 123-158.
McQueen, M.M. and G.T. Toussaint (1985). On the ultimate convex hull algorithm in practice. *Pattern Recognition Letters 3*, 29-34.
Overmars, M.H. and J. van Leeuwen (1981). Maintenance of configurations in the plane. *J. Comput. and Syst. Sci. 23*, 166-204.
Preparata, F.P. and S.J. Hong (1977). Convex Hulls of finite sets of points in two and three dimensions. *Comm. ACM 20*, 87-93.
Preparata, F.P. and M.I. Shamos (1985. *Computational Geometry An Introduction*. Springer-Verlag, New York, N.Y.
Raynaud, H. (1970). Sur l'enveloppe convexe des nuages des points aleatoires dans R^n, I. *J. App. Prob. 7*, 35-48.
Renyi, A. and R. Sulanke (1963). Ueber die konvexe Hulle von n zufallig gewahlten Punkten I. *Z. Wahrschein. 2*, 75-84.
Seidel, R. (1981). A convex hull algorithm optimal for points in even dimensions. M.Sc. Thesis, Tech. Rep. 81-14, Dept. of Comp. Sci., University of British Columbia, Vancouver, B.C.
Stewart, W.M. (1987). Three new algorithms for the d-dimensional convex hull problem. M.Sc. (C.S.) Thesis, School of Comp. Sci., University of New Brunswick, Fredericton, Canada.
Swart, G. (1985). "Finding the convex hull facet by facet. *J. Algorithms 6*, 17-48.

Discussion

Question: Is linear programming a *k*-dimensional problem with gift wrapping as one of the techniques used?

Answer: It is a dual problem. The simplex method proceeds from vertex to vertex, whereas the gift wrapping technique proceeds from face to face.

Question: Is the convex hull method a subset of the Voronoi diagram and is it a relatively efficient idea?

Answer: Both algorithms are order $n \log n$ time. We can also calculate the convex hull from the Voronoi diagram. However, it is more efficient to compute the convex hull directly.

Remark: By performing a boundary operation on the Voronoi diagram, we can obtain the convex hull? A point is on the convex hull if and only if the point's area is infinite.

Remark: To calculate the Voronoi diagram takes order $n \log n$ time. However, for a certain implementation, a less efficient on-line algorithm is good enough. The insertion process (point by point) is about one and a half times longer than the input/output process and hence the theoretical best algorithm is not needed.

Remark: Finding the starting point is a problem and a hierarchial search may be efficient for a large network, i.e., probably a balanced tree search.

Remark: Fractional cascading, a data structure by Chazelle and Guibas [1986] (*Algorithmica*, Vol. 1, pp. 133-192) may be the solution to our problem.

APPENDIX A

LIST OF PARTICIPANTS

Name and Address	BITNET Address
David Armstrong Topographical Engineering Topographical Survey Division Surveys and Mapping Branch 615 Booth Street Room 755 Ottawa, Ontario Canada K1A 0E9	
Yvan Bédard University of Laval, Dép. sciences géodesiques et télédétection, Laval, Quebec, Canada G1K 7P4	1130025@LAVALVM1
Claud Brunelle University of Laval, Dép. sciences géodesiques et télédétection, Laval, Quebec, Canada G1K 7P4	1130025@LAVALVM1
Cheng San Tan Department of Surveying Engineering University of New Brunswick P.O. Box 4400 Fredericton, N.B. Canada E3B 5A3	9069741@UNBMVS1
Dave Coleman Department of Surveying Engineering University of New Brunswick P.O. Box 4400 Fredericton, N.B. Canada E3B 5A3	9056454 @ UNBMVS1
Roger Dick Department of Surveying Engineering University of New Brunswick P.O. Box 4400 Fredericton, N.B. Canada E3B 5A3	9105501@UNBMVS1

Max Egenhofer
University of Maine
Department of Civil Engineering
103 Boardman Hall
Orono, ME 04469
U.S.A. MAX@MECAN1

Innocent Ezigbalike
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3 9047943@UNBMVS1

Andrew Frank
Surveying Engineering Program
University of Maine
Orono, ME 04469
U.S.A. FRANK@MECAN1

Chris Gold
Memorial University of Newfoundland
St. John's, Newfoundland
Canada. A1B 3X9 CGOLD @MUN

Ian Greasley
University of Maine
Orono, ME 04469
U.S.A. IAN@MECAN1

Mazlan Hashim
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3 9041259 @ UNBMVS1

Joe Horton
School of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3 JDH@UNBMVS1

Doug Hudson
University of Maine
Orono, ME 04469
U.S.A. HUDSON@MECAN1

Jeffrey Jackson
University of Maine
Orono, ME 04469
U.S.A. SVE36107@MECAN1

Marinos Kavouras
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3

Y.C. Lee
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B. LEE@UNBMVS1
Canada E3B 5A3 or SE@UNBMVS1

Jimmy Loh
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3. 9055692@UNBMVS1

Bob Maher
College of Geographic Sciences
Lawrencetown, N.S., Canada COGS@ACADIA

Brad Nickerson
School of Computer Science,
Univ. of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3 BGN@UNBMVS1.BITNET

Francois Paquette
Laval University
Dép. sciences géodesiques et télédétection,
Laval, Quebec, Canada G1K 7P4 1130025@LAVALVM1

David Pullar
Surveying Engineering Program
University of Maine
Orono, ME 04469
U.S.A. PULLAR@MECAN1

Vince Robinson
Department of Surveying Engineering
The University of Calgary
2500 University Drive NW
Calgary, Alberta
Canada T2N 1N4 VBROBINS@UNCAMULT

Matthias Uhlenbruck
Department of Surveying Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3

Mike Zhao
University of Maine
Orono, ME 04469
U.S.A. HUDSON@MECAN1