Experiments with a Rule-Based System for Interpreting Linear Map Features

Tony Schenk and *Ofer Zilberstein*
Department of Geodetic Science and Surveying, 1958 Neil Ave., The Ohio State University, Columbus, OH 43210-1247

ABSTRACT: In this paper we report about a research project which examined the feasibility as well as the applicability of building a system that controls the interpretation of linear features in digitized topographic maps. We developed a prototype system based on OPS5, a rule-based programming language. The control strategy of the inference process is influenced by control knowledge. Because the rules we used emphasize relations between objects rather than properties of a specific object, the system becomes task-driven. Results obtained from a section of a USGS quadrangle map are encouraging, and future work will involve refining the prototype as well as increasing the complexity by adding more objects.

INTRODUCTION

USING GEOGRAPHIC INFORMATION SYSTEMS (GIS) provides assistance in analyzing spatially related data with a large database and a query language. A growing number of organizations dealing with information related to map data are using GIS. Before one can take advantage of a GIS, however, massive amounts of data that constitute the foundation of the database must be entered into the system. Generating the database often presents a severe bottleneck and delays the operational use of GIS. Moreover, it represents 60 percent to 70 percent of the total project cost.

Information needed in the database may come from existing maps or from a data acquisition process, such as photogrammetry or field surveys. There are three different methods used to digitize existing maps. In the first, the map is placed on a digitizing table and an operator digitizes points, lines, areas, and symbols manually with a hand-held cursor. Before a map feature is digitized, information about its nature, e.g., feature, line, or symbol code, is entered. Manual digitizing is a tedious, labor consuming task and therefore an expensive process, but offers the advantage that the data are properly coded, allowing a topological structure in the spatial data base.

In the next category, the maps are digitized semi-automatically, using line-following techniques. An operator is still required to define the beginning of a line and to guide the system when it fails at intersections or otherwise gets confused. This method is superior to manual digitizing, particularly in the case of natural lines.

Finally, maps are digitized automatically with the help of scanning systems, which are also referred to as raster digitizing systems. The time to scan a map mainly depends on the size of the map and the required accuracy, and is independent of the complexity. It is generally agreed that scanning systems are the key to quicker and more economical data acquisition. The net result is a map that is converted to pixels, the size of which may vary depending on the accuracy selected. The only explicit information associated with pixels is their location; the semantic component is missing, that is, pixels have no label as to what feature or even object they belong to.

Usually, raster representation is transformed to vector representation. This format still leaves the burden of interpreting and editing to an operator, even though some state-of-the-art systems offer some basic interpretation capabilities based on pattern recognition methods.

Map interpretation belongs to a class of problems that defy conventional computer methods. In this paper we report about a research project that examined the feasibility, as well as the applicability, of building a system that controls feature detection in a map or image using Artificial Intelligence (AI) methods. For a comprehensive report the reader is referred to Zilberstein (1989). We focused on designing and developing a prototype for automatic tagging and interpreting linear features such as contour lines, streams, rivers, and roads that were digitized from a topographic map. The interpretation is performed using relations between the digitized segments, for generating an initial hypothesis, as well as geometric properties, for improving and complementing the initial hypothesis.

We used OPS5, a rule-based expert system shell, because it is suitable for using rapid prototyping methodology. Specifically, the prototype should shed light on the following questions:

- is map interpretation a feasible application?
- are rule-based systems adequate for this application; for example, is OPS5 the right AI language?
- what degree of automation can be expected?
- how should the knowledge and the control be structured?

In the next section we provide the reader with some background information about OPS5 and the problem domain. Then we describe the design and development of the prototype system followed by the results and conclusions of the research.

BACKGROUND

OPS5

In this section we present a concise summary about the expert system shell OPS5, which we used for prototyping the map interpretation application. Readers interested in more details are referred to Brownston (1986).

OPS5 consists of three major components:

Production Memory: collection of IF-THEN rules (also called rule base). Figure 1 is an example of a rule.

Working Memory: data representing the state of problem-solving. Rules match patterns (data) and transform them into new patterns.

Rule Interpreter: or inference engine runs the program. It matches rules with data and chooses the rule to be applied at each step.

Although OPS5 inherently executes rules in a forward-chaining fashion, it is possible to program backward-chaining solutions. The nature of map interpretation calls for backward-chaining: At the initial state we have many facts and the solution is found by putting forward a hypothesis that must be confirmed by finding supporting facts in the data base.

The working memory is structured into *element classes*. An individual element in the element class is called *working memory element* (WME). WMEs are defined by the class name and attri-

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 56, No. 6, June 1990, pp. 911–917.

0099-1112/90/5606–911$02.25/0

```
0632  ; English version of GSE:open-seg::seed
0633  ;  IF       there is a task to generate a seed element
0634  ;  and   there is an unknown open segment with max length and max
0635  ;           number of intersections
0636  ;  and   there is no element represents this segment as a seed
0637  ;  and   there is no element represents this segment as a contour
0638  ;  THEN  make a seed element
0639  ;  and   modify the segment's status to active
0640  ;
0641  (P GSE::open_seg_1::seed
0642    { (Context          ^task      generate_seed
0643                        ^status active)                          <cntx> }
0644      (crit       ^crit_min_seed_length <msl> )
0645    { (segment          ^status new
0646                        ^name      <nseg1>
0647                        ^length { <len> > <msl> }
0648                        ^nnode     { <inter> > 2 }
0649                        ^hypothesis unknown
0650                        ^type      1)                                <seg>  }
0651      (node  ^vector <nseg1> <nseg3>
0652                        ^type meet)
0653    - (segment          ^name { <nseg2> <> <nseg1> }
0654                        ^status new
0655                        ^type 1
0656                        ^length > <len>
0657                        ^nnode     > <inter> )
0658    - (element          ^segment_name <nseg3>
0659                        ^id <ele>
0660                        ^hypothesis          seed)
0661  -->
0662    (BIND <elem> (genatom))
0663    (MAKE element
0664                        ^id                  <elem>
0665                        ^segment_name        <nseg1>
0666                        ^hypothesis          seed
0667                        ^status     active
0668                        ^confidence          n/a
0669                        ^rule_name           IUS::open_seg_1::seed)
0670    (CBIND <new>)
0671    (WRITE (crlf) (SUBSTR <new> id  segment_name) <len>
0672              (TABTO 15)
0673              (SUBSTR <new> rule_name rule_name))
0674    (MAKE relation
0675                        ^type      bound_to
0676                        ^vector    <elem>    seed)
0677    (MODIFY <seg>  ^status active)
0678    (MODIFY <cntx> ^task generate_seed
0679                        ^status active) )
```

FIG. 1. OPS5 rule for selecting seed elements.

butes describing either facts about the WME or containing status information assigned during program execution.

Rules are independent entities in OPS5. They consist of the left-hand side (LHS) and the right-hand side (RHS), separated by an arrow →. The LHS specifies the conditions under which the rule will fire, that is, will execute the actions specified in RHS. As shown in Figure 1, variables can be assigned to rules (enclosed in angle brackets); however, they cannot be passed to other rules. They are simply a matching criteria when the left-hand side is matched with WME. Predicate operators specify the matching conditions. A rule may be fired; for example, if the attribute "length" has a value less than the value of the corresponding working memory element. The rule in Figure 1 specifies the conditions for selecting seed elements. The result is shown in Figure 5.

The RHS of a rule may specify actions to do things like

- create new WMEs (*make* action)
- change values in WMEs (*modify* action)
- remove WMEs from working memory (*remove* action)
- print output (*write* action)
- assign values to RHS (*bind* action)
- terminate process (*halt* action)
- call external subroutines (*call* action)
- build new rules (*build* action)

The rule interpreter executes the rule-based program by looping through the *recognize-act cycle*, involving the three steps of

(1) Match: during this process, the LHS of all rules are compared with WMEs. Successful matches (*instantiations*) are collected in the *conflit set*.
(2) Select: during the process of *conflict resolution*, one rule of the conflict set is chosen. Many possibilities exist to select the rule to be fired (e.g., first rule found). In OPS5, the *conflict resolution strategy* is based on the criteria of refraction, recency, and specificity. The user can select between two strategies: the lexicographic-sort strategy (LEX) and the means-ends-analysis (MEA).
(3) Act: the rule selected is executed and the working memory is updated.

Problem Domain

Maps as Graphic Documents. Maps are basically line drawings that convey information through shape, size, and relations among them. A topographic map portrays a simplified picture of reality. It represents spatial objects, along with their attributes, and relations according to the mapmaker's decisions during the mapping process. All sorts of topological and metric properties can be derived and identified. For example, relationships such as inclusiveness, nearness, close-to, and parallel can be defined.

Today, maps are increasingly stored in digital form (*digital map*). Many different formats are being used. Almost every digital mapping system has its own format with the ability to import files from another system by conversion routines. The amount of information that is stored in a topographic map is on the order of a few megabytes of data.

A map is composed of different classes of features, such as points (e.g., symbols), lines, and areas. In this work we concentrate on linear features because their interpretation is the crux of the automatic map interpretation problem. Points, depicted as symbols, appear in a unique way, and most of the time there is no ambiguity between them. Thus, their classification lends itself to algorithmic solution and there is no need to use knowledge-based techniques.

Because a map represents an interpreted scene, different linear features can be depicted by using different colors, styles, patterns, and widths (attributes of the linear feature). The difference between an interpreted line and the same line in reality is that the former one is the output of a generalization process. Moreover, on a map features are depicted based on human interpretation and the cartographic process.

Attributes greatly facilitate map reading. To deal with a sufficiently complex problem domain, we did not take advantage of attributes. That is, the test area is conceptually a black-and-white map with lines of equal appearance.

Entities, Objects, and Relationships. To describe a document that portrays reality in one way or another, one needs to formulate data models that are simplified views of part of the reality complying with certain rules.

Youngmann (1979) distinguishes two cartographic objects: primitive objects (e.g., point, line, polygon) and compound objects, which are collections of primitives (e.g., road-net). Cartographic objects are characterized by attributes and relations.

A relationship is the association between two types of objects; this can be syntactic (structure) or semantic (meaning). The cardinality of a relationship is the number of times the relation can occur between two specific objects (Bedard, 1989). Suppose a road crosses a river a minimum of zero times and a maximum of four times; on the other hand, a river can be crossed by a minimum of 0 roads and a maximum of N. We constitute a relation to cross with cardinality of 0,4 in one direction and 0,N in the other direction. Cardinality is useful for designing rules. Because contour lines cannot intersect each other, their cardinality is zero.

Objects and relations are characterized by attributes. When an attribute is used to identify a particular object or group of objects, it is named an identifier. For example, a contour is an identifier of a particular segment and elevation is the attribute attached to the contour.

We used the entity-relation diagram for expressing the semantics of data used in the test. A river, for example, is composed of two river_side elements with some measure of parallelism. A river_side element may have one or more streams connected to it on the same side with a node type meet. It cannot be crossed by a closed contour.

DESIGN AND DEVELOPMENT OF PROTOTYPE SYSTEM

In this section we describe the design of a prototype system. Because the application of "map interpretation" is partially ill-defined and inherently non-algorithmic, a structured software development approach is nearly impossible. Instead, we have chosen the rapid-prototyping methodology. The prototype is developed iteratively and serves to gain more insights into the problem domain, which in turn will subsequently lead to a more rigid program design necessary for the development of a production-level system.

The prototype system consists of input, output, inference engine, rules, and utilities. OPS5 is a data-driven system that provides no built-in constructs for control. We adopted a top-down strategy for controlling the inference process. In this way the system is driven by tasks (goals) rather than by data occurrences.

At every stage the system performs a *context task*. As long as there are candidates that comply with a specific context, OPS5 determines a conflict set, selects the proper element, and executes the specified action (recognize-act cycle). Then the next context is selected and the cycle starts again.

The inference is based on relations between linear features, which we call *segments*. The first step is to generate working elements for the known segments, such as index contours. The system then searches for the longest objects with a maximum number of intersections (*seed elements*). The seeds are assigned a hypothesis. The regions of interpretation around the seeds are increased using relations, such as segment-meet-segment. Geometric and consistency tests improve each hypothesis.

WORKING MEMORY ELEMENTS

The basic elements in the working memory involve

Segment: Linear entity consisting of one or more arcs. Segments are logical connections of arcs to a linear feature. A collection of segments form the data-base for the interpreter. Segments are class elements containing information (attributes), such as shape, length, etc. (see Figure 2). Each segment is interpreted by generating an element with an hypothesis. In some cases multiple hypotheses may be generated for the same segment.

Text: Entity resulting from character recognition process. Encouraged by recent advances in a related project we have made the assumption in this work that text is recognized and in fact available as data (see Boyer (1989)). Text elements are part of the data base.

Node:Each intersection between arcs is represented as a node. Nodes are the output of a routine that constructs the topology from arcs.

Relation: A working element expressing relations between working elements, such as inside, between, etc..

Figure 3 illustrates the concept of segments, nodes and relations as well as relations between objects.

KNOWLEDGE ACQUISITION

Apart from general common-sense knowledge, a map reader applies specific domain knowledge for extracting useful information from the map. We based our experiments on the

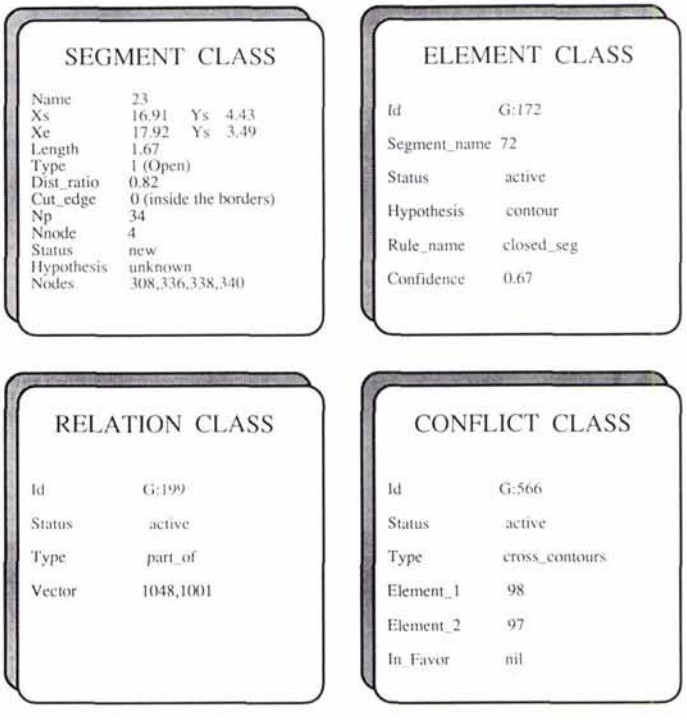

FIG. 2. Examples of working memory elements (WME).

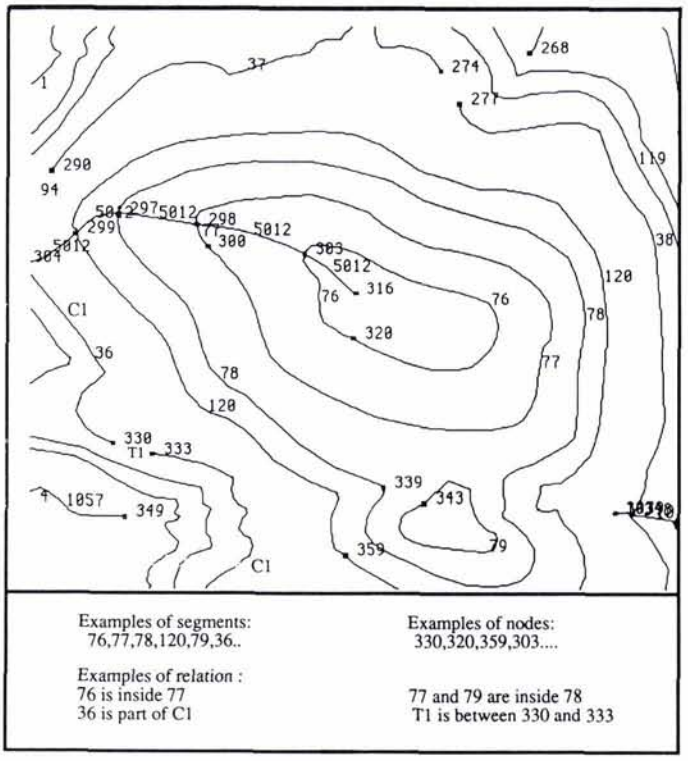

FIG. 3. Examples of segments, nodes, relations and relations between objects.

premise that the domain knowledge (*map interpretation knowledge*) is essentially a collection of rules.

The rules are basically divided into two categories. The first category consists of the global rules or long term memory (LTM). An example is contour lines. No matter what type of topographical map is used, contours cannot cross each other. The rules in the second category (STM short term memory) are related to a specific topographic map and thus may be different from document to document. For example, the index contour can be broken to accommodate the text or the text can be printed on the line.

Rules are also classified according to the type of knowledge they reflect. For example, one group may deal with cartographic procedures, while another group involves rules that express geomorphological phenomena. Some of the geomorphological relationships apply for a rather restricted area requiring special attention when implementing them.

Also, some rules must be as specific as possible for the system to cope with all kinds of peculiar situations. There are some swamps in the test area (see section *Experimental Results*), causing streams to start and to end with no connections. Generally, the system expects to find a stream network structure representing the water bodies (e.g., nodes from type meet). Thus, it will reject the hypothesis of a stream in the case mentioned above; therefore, a special rule (STM) must be added.

Overall Strategy for Map Interpretation Process

The general concept is governed by a top-down process, resulting in a task-oriented system as opposed to a data-driven system. The top-down strategy leads to a more restricted path in the search space; hence, it will converge faster to a solution. Interpretation is performed in a coarse-to-fine manner. The strategy is based on general knowledge about geomorphology and about relations between geographic objects.

The inference process is influenced by control knowledge. Because the rules express relations between objects rather than properties of a specific object, the inference process becomes task-oriented. First, rules are applied that are based on the relations between objects. Only after having an initial hypothesis does the system apply different tests which consist of geometric properties as well as specific relations. If two objects have the same relational structure, or if there is not enough relational information on a particular object, the system applies some property test such as a geometric test to obtain the initial hypothesis.

During the interpretation process, the system applies tests to selected features that might help in resolving conflicts between two elements. If a segment has high confidence, such as 0.7, it may be useless to apply a test that will improve the confidence only by a marginal amount. From an "intelligent system" one expects that it applies selective tests.

The interpretation process is divided into the following stages:

- Building working elements for known objects.
- Checking and interpreting segments with text.
- Generating special segments and major seeds.
- Checking local consistency on the interpreted elements.
- Interpreting special structures.
- Generating models.
- Resolving conflicts between models.
- Printing the final hypothesis with its confidence.

Steps that are concerned with checking and testing conflicts, such as the fourth and seventh steps, are not restricted to a specific task. They are fired by the inference engine whenever they match with working memory elements.

Uncertainties

In some knowledge-based systems, uncertainties are represented by using numerical techniques. Each uncertain fact has a numeric value representing the weight or the degree of confidence for the particular event. For the prototype we used Bayes' rule for calculating the outcome probability of an event, based on evidences and prior and conditional probabilities. The equation is given below:

$$P(H/E) = \frac{P(E/H)P(H)}{P(E)}$$

with

$$P(E) = P(E/H)P(H) + P(E/H)P(\neg H)$$

where H is the hypothesis and E the the evidence. Suppose $P(H)$ is the probability that a line segment is a contour line. Let $P(E/H)$ be the probability that the segment has a linearity measure of X given that it is a contour line and $P(E/\neg H)$ the probability that the segment has linearity X given it is not a contour line. Now a segment may have assigned the probabilities $P(H) = 0.6$, $P(E/H) = 0.8$, *and* $P(E/\neg H) = 0.2$. With the equation above we obtain $P(H/E) = 0.857$. The hypothesis that the segment is a contour line is improved from 0.6 to 0.857 because of the geometric property of linearity.

In general, the confidence interval is between 0 and 1. The higher value represents a higher degree of confidence. If two elements have the same interpretation, the system generates a conflict element or at the final stage selects the one that has the higher confidence value.

Tasks Description

This section provides an overview of tasks built into the system for conducting the experiments.

Task A: The system represents each known segment as a working element from the class segment having a known interpretation with confidence 1.0. The known segments (arcs with text in between) are produced in the

preprocessing stage and they serve as an anchor for the following steps.

Task B: The system checks and generates working elements from the element class contours based on text-segment relation. The system uses basic knowledge on the nature of contours. For example, contours are segments which can be closed, open on the edges, or open inside the map for index text (elevation). Contours do not intersect contours and they cannot be open inside the map. The task is to check whether the preprocess task produced consistent information. Currently, the system copes with open segments that are part of an appended segment with text between two segments. In this case, an element with a contour hypothesis is generated. Additionally, segments with no cut_edge and a text are assigned a contour hypothesis, which modifies the segment's type to *closed*.

Task C: The system selects seed segments based on the length and number of nodes. The selection complies with external criteria (e.g., no node type meet with another seed element) resulting in a skeleton structure guiding the next steps.

Task D: Here the system generates a working memory element for each segment that meets a seed element. The hypothesis is inherited from the seed, so if the seed is a stream the system generates a stream element. This task is executed whenever there is a match in the working memory. For example, whenever there is a seed element and a segment with node type meet , the system fires Task D.

Task E: Segments with some spatial characteristics are selected. The interpretation starts with segments that have special relations, as well as geometric characteristics. For example, a closed segment with no intersections is probably not a stream or a road. However, it might be a contour line or a lake. Now, the system generates one element for each interpretation with a predefined confidence. This means that, for a particular closed segment with no intersection, the system produces elements with different hypotheses and confidences.

Task F: After finishing the first stage of the basic interpretation of the major segments, the system applies different consistency checks to improve and refine the confidence of the interpretation. The consistency rules are basically a set of internal/external functions that produce more information on relations between elements or geometric properties of a specific segment. The relation two_side may serve as an example. It occurs when two segments are inside one segment and can be used to improve the confidence of being a contour. At this stage some rules activate external FORTRAN routines to supply information to be added to the database. The following predicate activates a function to determine whether all segments are on one side:

If all segments meet_one_side (segment_name) **Then** . . .where *meet_one_side* is a function that accepts the segment name as input and returns true/false.

Task G: Conflicts are dealt with between models. A stream must intersect contour lines in some logical order. For example, when moving along the stream (regardless of up or down) we would expect that elevations of intersected contour lines are monotonically increasing or decreasing. In case a model fails in a conflict test, no additional points are added to its confidence attribute. The same structure might have another interpretation; therefore, the most promising one will end up with the higher confidence.

Task H: In the final step the system resolves the remaining conflicts by selecting the hypothesis with the higher confidence. The output consists of a list of the segments and models with their confidences.

EXPERIMENTAL RESULTS

We present results of an area that we digitized manually from the USGS 7.5- × 15-minute quadrangle map Kempshall Mtn., New York, scale 1 : 25,000 (see Figure 4). We used the ARC/INFO system to perform the following tasks:

(1) Digitize the map using the module ARCEDIT. The features consist of lines and curves, called arcs in ARC/INFO.
(2) Define the topology (basically given by the intersections of arcs).
(3) Prepare a list of arcs and coordinates of all intersections.

The following steps then generated the data files containing the working elements for OPS5:

(1) Build a list of segments from arcs with the same identifier.
(2) Generate nodes as intersections of lines and assign the attributes *meet, cross, end,* and *ps*.
(3) Prepare a file of nodes, where each node contains a node name and a list of segments associated to that node.
(4) Digitized text is added as rectangles. Each text item is represented as a working element containing the type, location, etc..
(5) Prepare a list of segments that satisfy the criteria of contour lines.
(6) Prepare a file with relations among the different data types.

The preparation phase converts the digitized data (arcs and texts) into nodes, segments, and relations. The data base comprises 347 nodes, 214 segments, 152 relations, 55 text elements, and 11 contours.

After building the working memory and confirming the consistency between the known segments, the system starts with selecting the seed segments. Figure 5 presents elements that have been selected by rule IUS:OPEN:SEED. The selection criteria involve the length and number of intersections with other segments.

In the next step the seeds are assigned a hypothesis (either contour, stream, or road) by applying relation tests such as counting the number of intersections of type cross and the number of nodes of type meet. At this stage, a segment may be represented by more than one element. The system solves this conflict situation by assigning a conflict element with a unique identification number. If a seed was assigned the hypothesis stream, but has intersections of type meet with other segments that are all on the same side, then the system modifies the hypothesis to river_side.

After resolving all conflicts, each segment is assigned one hypothesis depending on the element with the highest confidence level. From a total of 214 segments, 15 were wrongly interpreted. The misinterpretations can be divided into three groups:

- Streams/lakes interpreted as contours (nine segments). In this group we find all closed segments that do not have intersections. The system assigns contour lines because this interpretation is more likely than shorelines. External knowledge on a specific region can provide more information. For example, if the region is a desert, then the probability of finding closed segments representing lakes is quite unlikely. Without additional information, however, the system cannot resolve this ambiguity. Another ambiguity may occur in the case of river banks without intersections of streams. Contour lines satisfy the same criteria.
- Streams interpreted as roads (five segments). Segments that start on the edge without meeting other segments are interpreted as roads. Again, the system lacks information for resolving the correct interpretation.
- Roads interpreted as streams (one segment). One road segment has been interpreted as a stream.

CONCLUSIONS AND FUTURE WORK

In this paper we described the development of a prototype system to intepret topographic maps. The results obtained from a test area (section of a USGS quadrangle map) are encouraging, and future work will involve refining the prototype as well as increasing the complexity by adding more objects.

The combination of a rule-based technique as a control and external routines for executing different tests appears to be feasible to solve the problem. False or ambiguous interpretations can be attributed to the limited knowledge currently encapsulated into the prototype. Situations like indistinguishable segments might be resolved by adding more specific rules and more sensitive criteria. Presently, the system uses only infor-

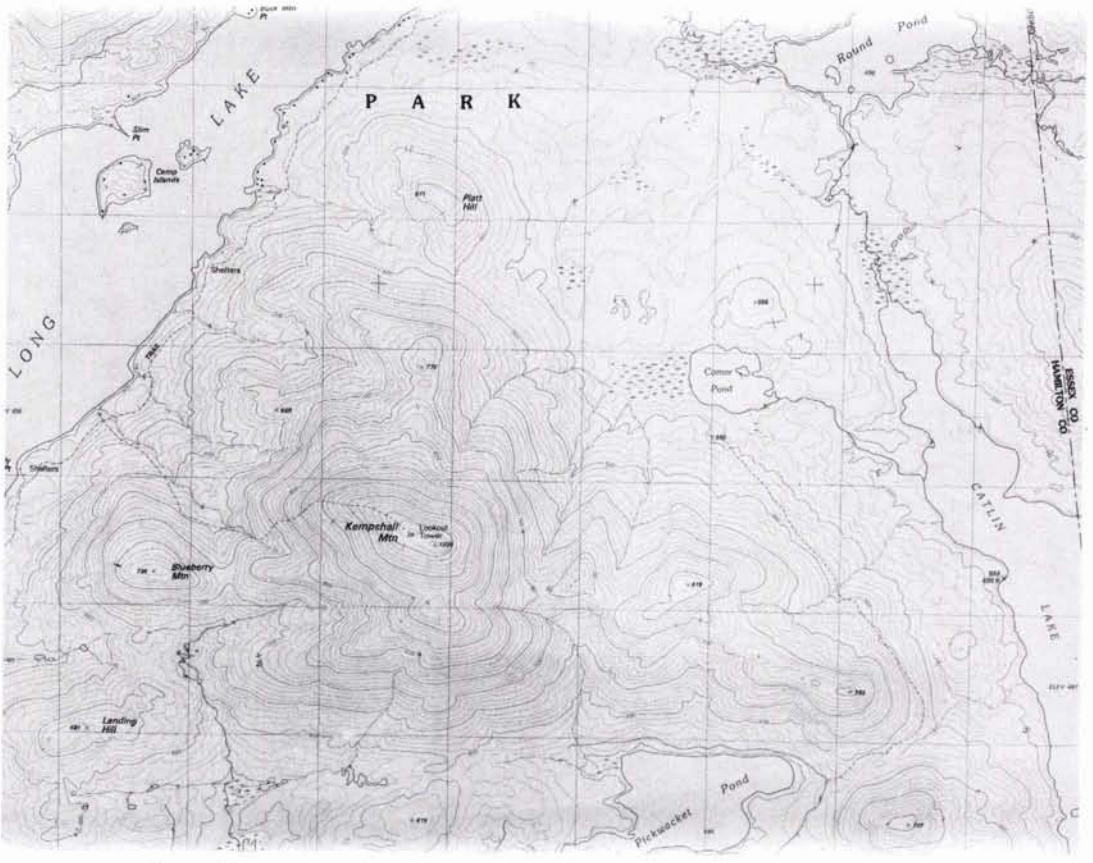

FIG. 4. Test area: section of USGS quadrangle map Kempshall Mtn., New York.

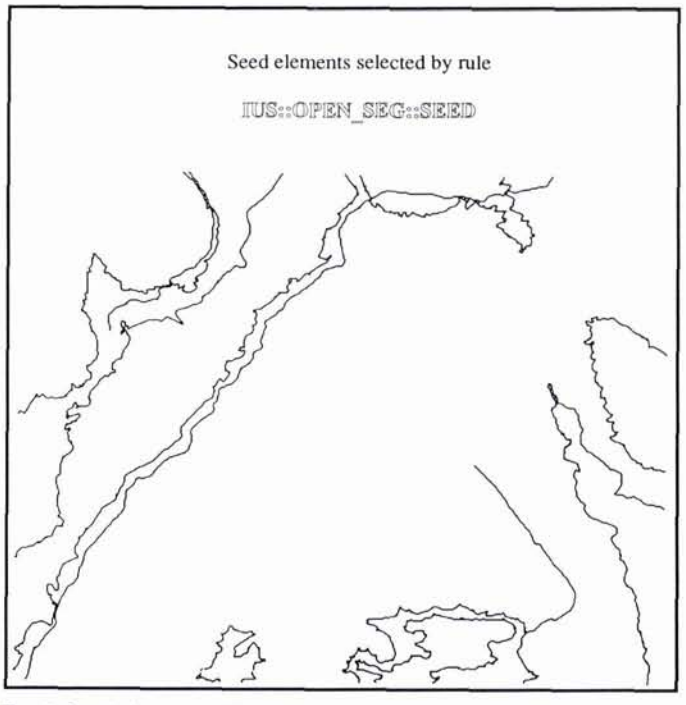

FIG. 5. Seed elements with segments selected by the rule shown in Figure 1.

mation about relations and geometric properties and it cannot resolve ambiguities as they may occur between, say, a pond and a small, closed contour line. Additional information, such as texture (area patterns), must be added.

The approach of using in the first step relations and topology between features, and only then using geometric properties, seems to be very promising. Relations are well defined; moreover, *a priori* knowledge on some of the relations between features makes it possible to identify possible or impossible situations.

During the inference the system generates relations (e.g., part_of). They are stored as working memory elements and thus build a semantic network that offers the advantage of formulating queries.

ACKNOWLEDGMENTS

Funding for this research was provided in part by the NASA Center for the Commercial Development of Space Component of the Center for Mapping at The Ohio State University. The authors would like to thank the reviewers whose valuable comments were greatly appreciated.

REFERENCES

Bedard, Y., 1989. Extending entity/relationship formalism for spatial information systems. *Proceedings, Auto-Carto 9*, 2–7 April, Baltimore, Maryland.

Boyer, K.L., 1988. *Feature Extraction from Aerial Photographs and Interpreting Raster-Scanned Map Data*. Annual Report NASA-Project, Center for Mapping, The Ohio State University.

Brownston, L., *et al.*, 1985. *Programing Expert Systems in OPS5*. Addison-Wesley, Reading, Mass.

Youngmann, C.E., (1978). *A Linguistic Approach to Map Description*. Harvard Papers on Geographic Information Systems, Vol. 4 (E. Dutton, ed.), Harvard University Laboratory for Computer Graphics and Spatial Analysis, Cambridge, Mass.

Zilberstein, O., 1989. *Studies in the Use of a Rule-Based System for Linear Features Extraction from Digitized Topographic Maps*. Techn. Rep. No. 456, Dept. of Geodetic Science and Surveying, The Ohio State University.

(Accepted 18 January 1990)