The ISR: A Database for Symbolic Processing in Computer Vision

J. Brolio, B. Draper
J.R. Beveridge and A. Hanson

COINS TR 89–111

November 1989

The ISR: A Database for Symbolic Processing in Computer Vision*

John Brolio, Bruce A. Draper, J. Ross Beveridge, and Allen R. Hanson
Computer and Information Science Department
University of Massachusetts, Amherst

August 26, 1989

Abstract

Computer vision imposes unique requirements on the representation and manipulation of image data and knowledge. The interpretation of an image can generate thousands of intermediate level descriptions, many of which must be repeatedly accessed and processed. Traditional knowledge representation methods do not provide mechanisms to accomplish this effectively. We describe a database management system called the Intermediate Symbolic Representation (ISR) which is suitable for use at the intermediate (symbolic) level of vision. The ISR, which is based on database management methodology, mediates access to massive quantities of intermediate level vision data, and forms an active interface to the higher level inference processes responsible for constructing the interpretation of an image. We also present several examples illustrating the use of the ISR.

1. Introduction

The VISIONS Image Understanding System is a research environment within which general theories of vision can be designed, implemented, and tested. This paper is concerned

*This research has been supported in part by DARPA under contract #F30602-87-C-0140 and NSF DCR-8500332.

with the design and implementation of a representation and management system (ISR) for data intermediate between the purely numeric image arrays of digitized sensor input and the semantic world models which form the high-level interpretation of a scene. The system supports the types of data and operations that experience has shown to be important in vision and is flexible enough to adapt to the changing needs of ongoing research. Furthermore, it provides a centralized data representation which supports integration of results from multiple avenues of research into the overall vision system.

Section 2 briefly describes the computational paradigm for vision which motivates much of the underlying structure of the ISR. Section 3 specifies the minimal set of database requirements for intermediate-level vision which must be met by the system. It also discusses those aspects of the ISR which are similar to conventional databases and to knowledge representation languages from artificial intelligence. Section 4 describes some of the primitive data types and access functions of the ISR data management system, and Section 5 sketches several examples of how these primitives are used in intermediate-level vision applications.

2. A Computational Paradigm for Computer Vision

The goal of computer vision is to construct automatically from a digital image (or an image sequence if motion is involved) a symbolic interpretation which describes the environment from which the image was captured. This description will include, among other things, the identity and structure of objects and their spatial and temporal relationships.[1] The requirement that objects be identified implies that vision systems must have access to descriptions of the objects (or of generalized object classes) stored in a knowledge base in a form that allows them to be matched to image structures. Consequently, a generally

[1]Note that in this paper we restrict the discussion to static images, captured from a single sensor, for which there is no motion information.

accepted paradigm for computer vision involves multiple levels of representation and abstraction, from the purely numeric image representation to the highly abstracted object or object class models stored in the knowledge base [7].

This general paradigm underlies the VISIONS Image Understanding System, a research system being developed in the Computer Vision Laboratory at the University of Massachusetts. The VISIONS system is organized into three conceptual levels of visual processing: low (image level), intermediate, and high (knowledge level). At the low level of vision, potentially useful image events, such as homogeneous regions (collections of contiguous image data points with similar properties), edges and lines, 3D surface patches, motion information, etc. are extracted from the image data, typically without recourse to knowledge of the image contents.

Descriptions of these events are stored at the intermediate level as named and typed symbolic entities called tokens. Each token contains attribute-value pairs which describe the event, for example the color and texture of a region or the length and contrast of a line segment. Additional tokens and token types are created by grouping, splitting, and/or modifying existing tokens. Tokens can be constructed from other tokens, imposing a hierarchical structure on the representation.

High level knowledge is organized into object descriptions called schemas. Schemas are organized into relational networks, such as a part-subpart hierarchy, to facilitate recognition. Each schema has an associated procedural component which, when executed, searches for evidence of the object in the low- and intermediate-level data. Schemas are active entities and communicate asynchronously with each other by means of a global blackboard maintained by the schema system control shell.

Figure 1 is a stylized and highly simplified diagram of the three levels of representation in the VISIONS system. More detail on the VISIONS system, as well as image interpretation

results on house and road scenes, may be found in [3].

A major consideration in creating an interpretation is the inevitable mismatch between idealized models and bottom-up data. The output of even the best low-level algorithms must be considerably transformed prior to, or during, the matching process. In the VISIONS system much of the transformation takes place at the intermediate level. Perceptual organization algorithms split, merge, add and delete intermediate level tokens as well as create more abstract tokens from simpler ones. Examples of more abstract tokens include line groups which satisfy certain geometric properties and aggregates of regions and lines consistent with an object or object part in the knowledge base. Additional examples of aggregate tokens and the processes which create them are found in Sections 5.2–5.3. There is no practical limit to the variety or quantity of tokens which are based on aggregates of other tokens.

Tokens at the intermediate-level are not isolated events. Often their most important properties are their relations to other tokens, as in the line grouping algorithm of Boldt, et al. ([1]; [10]). This algorithm groups short lines to form longer ones in a recursive cycle of linking, grouping, and replacement according to criteria based on endpoint distance, relative orientation, and lateral displacement. The algorithm terminates when no further line segments can be replaced. At each recursive level old line tokens are retrieved by spatial proximity and orientation. Since an image of reasonable size and complexity may have 10,000 or more initial line tokens, the spatial relations at the heart of this and similar algorithms must be efficiently represented.

Experience has shown that the amount of intermediate level data involved in the interpretation of a single image is quite large. Table 1 shows the size and storage requirements for five types of symbolic image events (not including low level image data) extracted for nine typical images used in interpretation experiments. The event types are straight lines,

image	number of regions	number of straight lines** (short/other)	number of line*/region intersections	number of boundary lines** (short/other)	number of boundary line*/region intersections	data size including token features (in Mbytes)
ROAD1	187	3981/911	1517	1756/206	724	1.6
ROAD2	307	4062/963	1868	2811/212	751	2.0
ROAD5	427	4019/1051	2736	3733/379	1415	2.4
ROAD10	311	4209/996	2111	3246/242	853	2.1
ROAD16	222	3744/733	1112	1305/187	775	1.5
ROAD25	356	4364/949	2354	3730/285	980	2.3
HOUSE1	305	2843/878	1793	1174/405	1438	1.5
HOUSE7	168	3522/900	1823	2239/242	711	1.6
HOUSE10	165	2675/845	1255	965/241	836	1.2
*	*Line-region intersections and features are not calculated for short lines.*					
**	*Short lines are those whose length is less than 5 pixels.*					
Note:	*All images have a resolution of* 256 × 256 *pixels.*					

Table 1: Size of the Intermediate Symbolic Representation after Low-level Processing

regions, intersections of lines with regions, lines lying on region boundaries, and intersections of lines with region boundaries[2]. Note that lines shorter than 5 pixels are counted only where indicated. The data in Table 1 represents only image events (and relationships between them) that are initially extracted from the image, prior to execution of any intermediate level grouping process or schema interpretation strategy.

3. Database Requirements for Image Interpretation.

A database system for vision research must satisfy two distinct requirements: (1) it must provide efficient access to and manipulation of intermediate-level data, and (2) it must provide a simple and intuitive applications programming language for researchers who are programmers out of necessity. Efficiency here is more than just a matter of convenience. The computational burden of vision is such that certain experiments are not feasible with current

[2]Justifications for these events is beyond the scope of the paper; however, these and other similar events have been used in most of the interpretation experiments performed to date.

technology except with a highly optimized database. Consequently, the database should not incur overhead for features a researcher will not use. On the other hand, researchers will not use a database language that requires excessive learning or programming effort for basic functions. If the system is not generally used, data will not be shared, machine-readable data output from experiments will be effectively lost, and researchers will constantly re-implement the same data structures and manipulation procedures.

Therefore, we have designed the ISR as a "reduced instruction set" for applications in computer vision. The ISR provides a set of primitive operations and representations which are easy to use, and with which any intermediate-level applications can be built. We have given a great deal of attention to optimizing these basic building blocks.

3.1 Primary requirements

In designing the ISR, we have drawn from both database practice and knowledge representation (KR) technology in artificial intelligence (AI), the former for efficiency and generality and the latter for flexibility. Additionally, we have based our design on the experience of researchers in the VISIONS environment to determine an efficient set of language primitives for spatial retrieval applications. What follows is a summary of significant intermediate level data requirements. The first requirements address efficient support of the necessary data manipulation and retrieval operations.

Req.1: Vision research requires spatial datatypes and retrieval methods not usually supported in standard database systems. An algorithm like the Boldt line extractor (Section 2) may make hundreds of thousands of spatial access queries on a single image. The algorithms to be discussed in Sections 5.2 and 5.3 do less work in total, but the ratio of spatial queries to other database queries remains about the same. The spatial access functions which implement these retrievals must be efficiently implemented. The ISR supplies meth-

ods for retrieving all tokens which intersect a given bounding rectangle or all tokens which intersect a region of arbitrary shape. This feature is the main focus of the ISR design; we continue to research optimal methods of spatial data representation and retrieval.

Req.1: Data structures should be standardized sufficiently to allow efficient sharing of data: (a) incremental saves and loads of partial datasets must be possible; (b) data must be saved in logical modules of reasonable size and scope; (c) it should be possible to load selected records and selected fields of those records. All of these requirements are fulfilled by Database Management System (DBMS) systems; knowledge representation systems do not currently provide much capability or flexibility in this area.

Req.2: The researcher must be able to reconfigure the database dynamically, to add fields or attributes to a record, or to create a new dataset with records or attributes from one or more existing datasets. Every researcher is free to alter the syntax, semantics and facts in a database; it should be easy to keep these changes private or to make them available to the research group as needed, without storing large quantities of redundant data. In the ISR, it is a simple matter to create a dataset, or alter its definition, or to copy, load or save a selection or projection of a dataset. The system provides reasonable means of tracking the sources of various elements of a user's individual database.

Req.3: Like any research tool, the database application language must be as powerful and as uncomplicated as possible. It must be fully embedded in the host language (Lisp or C, in our case) and must imitate the best features of that language as much as possible. To satisfy this requirement, a prototype of the ISR was built and put into use. The ISR design group then spent a year and a half analyzing information on the use of the prototype, spatial retrieval applications written or proposed for it, and requests for changes or new features. These were integrated into the current design.

Req.4: Extensibility must be built into the database. The design must be modular and open-ended, so that future developments are not precluded. Source code must be available. When a commercial DBMS is chosen and the databases are designed in it, the data requirements are expected to be fairly stable. In a research environment, however, future requirements for data and database operations cannot be fully known *a priori*. As research progresses, the requirements may change drastically. It is inevitable (and desirable) that new operations will be demanded. Primitive operations must be modular enough that the user can combine them to generate new functions.

3.2 Database technology

Vision research demands basic facilities for storing, sharing, accessing, selecting, and sorting large quantities of data. But there are some points where vision data requirements diverge from capabilities provided by a DBMS and a great deal can be learned about an ideal data management system for computer vision by examining those points.

The traditional DBMS environment demands three separate levels of human intervention: the database administration level, the applications programming level and the end-user level. These interactions are often mediated through two or three different languages in the database, as well as different levels of privilege. In a computer vision research environment, however, there is often no distinction between these three types of interaction. An individual researcher may create, delete and restructure a database, write complex experimental applications in the database language, and examine the results on a graphics screen. Embedding these three distinct levels of functionality in one efficient, conceptually simple language represented a major challenge for the design of the ISR.

3.3 Knowledge Representation Technology

In contrast, knowledge representation technology as it currently exists has some positive qualities, which are being incorporated into new database research[9], but it also has a number of deficiencies, some of which can be remedied by recourse to DBMS methods. The positive features are extensibility, flexibility and the capability for procedural attachment. We have borrowed these from KR technology in the concept of a *frame* as a datatype which can be redefined dynamically and *demons*, which are procedures which can be executed when data field values are accessed.

Unfortunately, such common database utilities as sorting or indexing have been quite rare in KR systems, perhaps on the presumption that knowledge needs a great deal of hierarchical structure and very little linear structure. Vision and much other AI research, however, requires full support of standard database operations such as sorting, selecting and indexing and of standard datatypes (e.g., arrays and sets in addition to integers, floats strings, etc.).

For many KR systems, data storage and retrieval are an afterthought. Saving and reloading datasets is usually a cumbersome operation. Data sharing is extremely difficult because data values and data descriptions are stored as a unit.

Built-in inference procedures in KR systems are generally wide of the mark from our point of view. In any area as data-intensive as computer vision, any data-driven processing must be carefully controlled. A weak method such as forward chaining must be used with a great deal of top-down control to restrict the generative effect of thousands of pieces of data. Furthermore, many computational decisions in vision processing are made with statistical or combinatorial optimization techniques, so that it will require some future extension of a constraint programming language to support the kind of mathematical inference required.

We examined current database theory and practice to determine if an appropriate sys-

tem already exists. Image database management systems do not focus on operations essential to computer vision research, where indexing objects to images is not a typical task, but finding the intersection of arbitrary subsets of pixels is very common. Engineering (CAD/CAM) database systems, although they require similar flexibility and extendability, are even less appropriate in functionality, since there is nothing equivalent to *low level* vision in CAD/CAM representation. What was required and not available was a system tailored to the needs of computer vision research, which is easily adaptable to different machines and languages.

In the computer vision field, there are other database systems, most notably CODGER [8] from Carnegie Mellon and CKS [5] being developed at SRI. CODGER has evolved over the same time period as the ISR and there are similarities even in nomenclature. But CODGER is broader in scope, encompassing very high-level representation and process scheduling in addition to intermediate level data management. Furthermore, CODGER appears to treat many functions as primitives which the ISR would view as application programs, such as transformation of 3D coordinates. Not all researchers working at the intermediate level need or want high-level representations or operations, and even high-level vision researchers may want a different methodology than the one supported. In the long run, such an all-encompassing approach may render the database system cumbersome and hence less likely to be used by the researchers for whom it is intended.

CKS is a more recent design which seems to be intended as a knowledge representation system for three-dimensional world modeling at a very high level. It devotes much attention to semantic issues which are less essential at the intermediate level of computer vision, and hence less attention to the questions of efficient spatial retrieval and manipulation of two-dimensional symbolic entities.

The ISR seems to be unique in its fulfillment of the needs for a small, efficient and

flexible database management system focussed on the needs of intermediate level vision. The next section describes the ISR, and the section following that provides examples of uses of the system in fairly typical computer vision research activities.

4. Description of the ISR Data Management System

The design of the ISR was constrained both by the nature of symbolic image interpretation and the need to serve multiple researchers with varied interests. A small set of highly optimized primitive commands and representations are provided. These can be combined to meet the needs of a particular user's research. For example, common structures such as line segments and regions, along with their associated operations, are not part of the system primitives, but are kept in optional libraries. As new token types and operations are found useful they may be placed in additional libraries. In this section we outline the primitive data objects in the ISR, and in the next section we sketch three examples of how these primitives are used in intermediate-level vision applications.

4.1 Tokens and Features

The fundamental object in the ISR is a *token*, which is similar to a record in a standard database management system. A token can be used to represent an image event like a line or a homogeneous region in an image, or an aggregate of events, such as a group of parallel lines, a geometric structure, or the regions and lines which are hypothesized to belong to some object. The data fields of an ISR token, which describe attributes of the referenced event, are called *features*. ISR features are similar to frame slots in a KR system, in that they have attached procedures (*demons*) which can be activated whenever a value is requested or modified, or when a value is needed which has not yet been computed. ISR tokens for similar image events (e.g. all the lines for one image) are grouped together into

a *tokenset*. Tokensets allow operations over similar data, such as displaying every line in a tokenset, or computing the contrast across each line.

4.2 Frames

Each tokenset is embedded in a descriptor object called an ISR *Frame*. The relationship between a frame and its tokenset is similar to the class/instance relationship found in object-oriented databases. Each token represents an event, such as a region extracted from an image, while the frame represents the class of events, in this case the set of all regions.

Frames, like tokens, are first-class data objects with features. Frames are linked into hierarchies which denote relationships between associated tokensets. For example two frames containing the lines and regions from a single image might both be children of the same image frame (see Figure 2). Additionally, the frame provides a modifiable description of its tokenset. For example, an ISR frame feature may contain statistics on tokenset feature values, such as the mean and standard deviation of the LENGTH feature for a set of lines. Since the most natural hierarchy for one problem may not fit another, the frame hierarchy may be specified by the user.

4.3 Feature Datatypes

A token or frame feature can have one of the following datatypes: integer, float, string, array, pixelmap, ISR handle, or pointer[3]. Pixelmaps are 2D bit-arrays that specify a set of pixels. The operations defined over pixelmaps are union, intersection, set-difference and a count function that returns the number of pixels. Pixelmaps are used to map regions (which have no simple, analytic form) onto images. For example, a pixelmap acts as a mask, specifying which pixels are in a region and should be summed to calculate the average

[3]The *pointer* datatype is used for storing non-portable data objects.

intensity of the region.

Since feature storage is allocated at runtime as needed, it is possible to create a virtual feature, whose value is computed whenever it is needed and never stored. This is useful for defining data aggregations which are constructed on demand from primitive features. For example, the endpoints of a straight line may be stored as four features, x_0, y_0, x_1, y_1. If a user program needs to access each endpoint as a vector, a virtual feature called **endpoint** can be created. **Endpoint** consumes no data storage space; when accessed, it constructs a vector out of x_i, y_i, which is returned to the caller. Similarly, a vector of the form x_i, y_i can be used to "set" the value of **endpoint**, in which case a storage demon breaks the vector up and stores each value in the appropriate field. Virtual features are also useful for implementing transparent conversions between different representations of the same data, such as polar/rectangular or ego-centered/world-centered coordinates.

4.4 Handles and Subsets

One of the fundamental requirements of the ISR was that it be able to express relations between tokens, and in particular associative relations. Simple pair-wise relations can be expressed through token features of type handle. A handle is a reference to a token or frame in the database. Associative relations are represented by *tokensubsets*. As the name implies, a tokensubset specifies a selection of some or all of the tokens in a tokenset. A tokensubset can be used to denote the set of all regions whose average intensity exceeds fifty grey levels, or all lines that are at least five pixels long. The ISR functions that create and manipulate tokensubsets can be viewed as a software analogue of content-addressable or associative memory. Tokensubsets can be 1) selected by numeric feature ranges or properties of pixelmaps, 2) combined by basic set operations such as union, intersection or set-difference, and 3) used as a guide to control function application. An example of the last case is computing

the contrast of every line greater than five pixels long. A special class of tokensubset, called a *sort*, represents an ordering over the tokensubsets elements. Tokensubsets and sorts are both handles, and can be stored on any token or frame feature of that type.

5. Example Uses of the ISR

5.1 Classification

One of the oldest and most studied problems in computer vision is the classification of image regions by feature values[4]. Color, texture, shape, location or other attributes of a region are used to categorize it as belonging to one of N classes. One approach to classification, which takes advantage of lazy evaluation, is based on *decision trees* (also called *regression trees*[2]).

In a decision tree every leaf node carries the name of a category, and every internal node selects a feature to be tested (see Figure 3). The classification procedure begins by making the root of the decision tree the "current node". The procedure then enters a loop in which the current node determines which feature of the region should be tested, and the resulting feature value dictates which child node should become the current node. The loop exits when the region reaches a leaf node, at which time the region is assigned to the category on the leaf node.

Decision trees are popular in part because of their efficiency. Unlike a Bayes classifier, a decision tree only requires that a few of a region's features be computed. To be precise, only the features on the path from the root node to the eventual leaf node need be computed. To take advantage of this the ISR supports lazy evaluation. When the ISR is used as the database for a decision tree, the region feature values need not be computed beforehand. Instead, the function for computing each feature of a region is installed as a demon. When classifying a region, the tree traversal algorithm asks for the value of the feature at the

current node. The demon notices that the value has not yet been computed, calculates it, and returns the appropriate value; the value can also be stored in case it is requested again. In this way, feature demons are used to avoid computing unneccessary feature values.

5.2 Perceptual Grouping

Grouping related tokens into aggregate structures (*perceptual grouping*) is a common intermediate vision task. One strategy groups tokens by the reflexive closure of one or more relations. If the tokens are viewed as nodes in a graph and the relations as adjoining arcs then this style of grouping forms the connected components of the graph. The result is a set of tokens which together represent a new image event, and which may possess properties not possessed by any of its components.

The Rectilinear Line Grouping System (RLGS; [6]) is a perceptual organization system implemented in the ISR. The basic relations measured by the RLGS are whether two proximal lines are colinear, parallel or perpendicular. For every line in the image, the RLGS finds the set of lines that are proximal and parallel to it, and stores this information as a tokensubset on the line token. This process is repeated looking for lines that are proximal and colinear or proximal and perpendicular. Finally, the RLGS looks for groups of lines connected by one or more relations (colinear, colinear or perpendicular, etc.). Figure 4 depicts such a line group. The group can either be returned as a tokensubset to the user, or stored as a feature on a "line group" token. In the latter case, other features of the line group, such as its minimum bounding rectangle, can also be computed and stored.

5.3 Spatial Access (Bigcells)

Spatial proximity is an important consideration in accessing image information. It is often desirable to access just those tokens lying on or near a particular point in the image.

To this end, grids are often imposed on the two-dimensional image, dividing the image into rectangular cells (*bigcells*). If lines, regions or other image events are stored according to the bigcells that they intersect, it becomes relatively easy to retrieve the neighbors of a token by accessing only those bigcells which lie within the radius of interest.

Bigcells have been implemented in the ISR by creating a tokenset in which each cell of the grid is represented by a token. Each feature of the cell token represents a subset of objects (such as lines or regions) which spatially intersect that cell. Functions have been written for storing tokens in the appropriate cells and for retrieval based on eight types of spatial relations (point-to-line, line-to-line, region-to-region, etc.).

A typical use of the spatial indexing grid occurs during the interpretation of a road, which is part of a larger interpretation effort involving a road scene [3]. Figure 5a (upper left) shows a photograph of a typical road scene. Figure 5b (upper right) shows the output of a low-level algorithm for identifying lines applied to the lower left quadrant of the image. The goal is to identify the boundary of a roadline from the set of lines identified by a low-level line extraction algorithm. The method is to construct, from existing line segments, a line-chain which satisfies the constraints for a centerline as represented in the knowledge base.

Although the algorithm described below could be applied to the inital set of lines derived from the entire image, typically the interpretation process would already have determined a context for the centerline [3], for example, by hypothesising the road surface or road sides. The context spatially constrains the possible locations of the centerline and reduces the combinatorics of the search process. The line-chain algorithm starts with a line which has been determined to be a good candidate for a roadline boundary line. That line is extended by joining it end-to-end with other lines which meet it near its endpoints. Candidates for extension are selected by retrieving any lines within a small radius of the endpoints of the

current line-chain. Figure 5c (lower left) shows (along with the final successful line-chain interpretation) the grid cells accessed during the search and the candidate lines retrieved. Figure 5d (lower right) shows the set of candidates after selecting only those within a small distance of the current line-chain endpoints. The resulting spatial structure will then be subject to further verification and validation by high level schema strategies before being added to the evolving overall interpretation.

Grids implemented in this way do not have to represent regular equal divisions; furthermore, a grid can partition not only spatial attribute values but any numeric attribute with a finite range. Access to tokens by any feature can be supported by dividing the feature into ranges and storing tokens as tokensets on the corresping cells. Thus the same primitives that implement spatial access can be used to implement histogram-based algorithms and generalized Hough transforms.

6. Conclusion

Experience with intermediate-level computer vision has taught us three lessons.

- Spatial representations, spatial accesss and procedural attachment are essential.
- A database for a research environment must be easy to use and easy to modify.
- Efficiency is critical.

The ISR combines the standard access and retrieval mechanisms of DBMS technologies with the notion of frames and demons found in knowledge representation languages. Most importantly, the ISR provides special spatial representations and acccess routines not traditionally found in either of these two fields.

Versions of the ISR have been operational in the Computer Vision Laboratory at the University of Masschusetts for over three years; the current version is implemented in LISP and has been in use for over a year. In addition, a version of the ISR has recently been

embedded in a commercially available system.

The major conclusion which can be reached from this effort is that neither traditional database management systems nor knowledge representation languages from artificial intelligence support the primitive spatial data structuring and access functions required at the intermediate-level of vision. The ISR adds spatial primitives to the capabilities available in DBMS and KR systems to provide a simple yet effective database for intermediate-level vision.

7. Acknowledgements

We are grateful to Bob Collins for feedback during the design of the ISR, for the development of the decision tree application, and for producing Figure 7.. We also gratefully acknowledge the help of Joey Griffith, Bob Heller, Ric Southwick and Jim Burrill in designing and implementing various versions of the ISR. In addition, we would like to thank the many members of the UMass computer vision laboratory who have used the ISR and given us invaluable feedback.

REFERENCES

[1] Boldt, M., R. Weiss and E. Riseman. (to appear). Token-Based Extraction of Straight Lines, IEEE-SMC.

[2] Leo Breiman, Jerome H. Freidman, Richard H. Olshen, and Charles J. Stone. *Classification and Regression Trees,* Wadsworth, Inc.: Belmont, CA., 1984.

[3] Bruce A. Draper, Robert T. Collins, John Brolio, Allen R. Hanson and Edward M. Riseman. "The Schema System", *International Journal of Computer Vision*, 2 (1989), pp. 209-250.

[4] R. O. Duda and Peter E. Hart, *Pattern Classification and Scene Analysis* 1973, John Wiley & Sons: New York.

[5] Martin A. Fischler and Robert C. Bolles, "Image Understanding Research at SRI International," Proceedings of the 1989 DARPA Image Understanding Workshop, Palo Alto.

[6] George Reynolds and J. Ross Beveridge. "Searching for Geometric Structure in Images of Natural Scenes", *Proc. of DARPA Image Understanding Workshop,,* Los Angeles, CA., pp. 257-271, February 1987.

[7] Azriel Rosenfeld. "Image Analysis: Problems, Progress and Prospects," *Pattern Recognition* 17(1):3-12, 1984. Also appears in *Readings in Computer Vision*, Fischler and Firschein, eds., Morgan-Kaufman: Los Altos, CA 1987, pp. 3-12.

[8] Steven A. Shafer, Anthony Stentz, Charles E. Thorpe. "An Architecture for Sensor Fusion in a Mobile Robot," Proc. IEEE Int'l. Conf. on Robotics and Automation, San Francisco, 1986.

[9] M. R. Stonebraker. "Object Management in POSTGRES Using Procedures", Proc. Int. Wkshp on Object-Oriented Database Systems, Asilomar, CA, Sep. 1986.

[10] Weiss, R. and M. Boldt. (1986). "Geometric Grouping Applied to Straight Lines," Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, pp. 489-495.

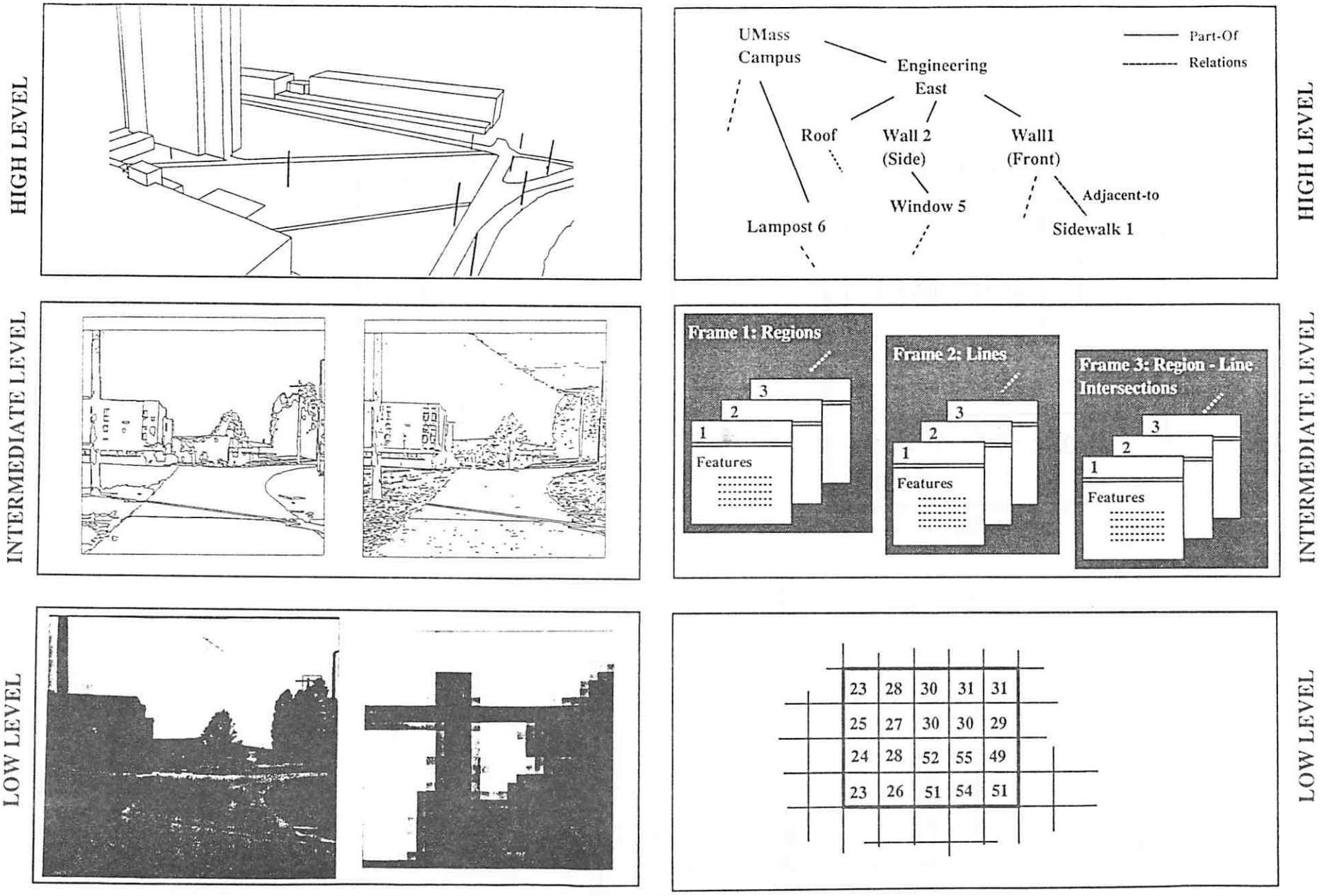


FIGURE 1

Brolio et al

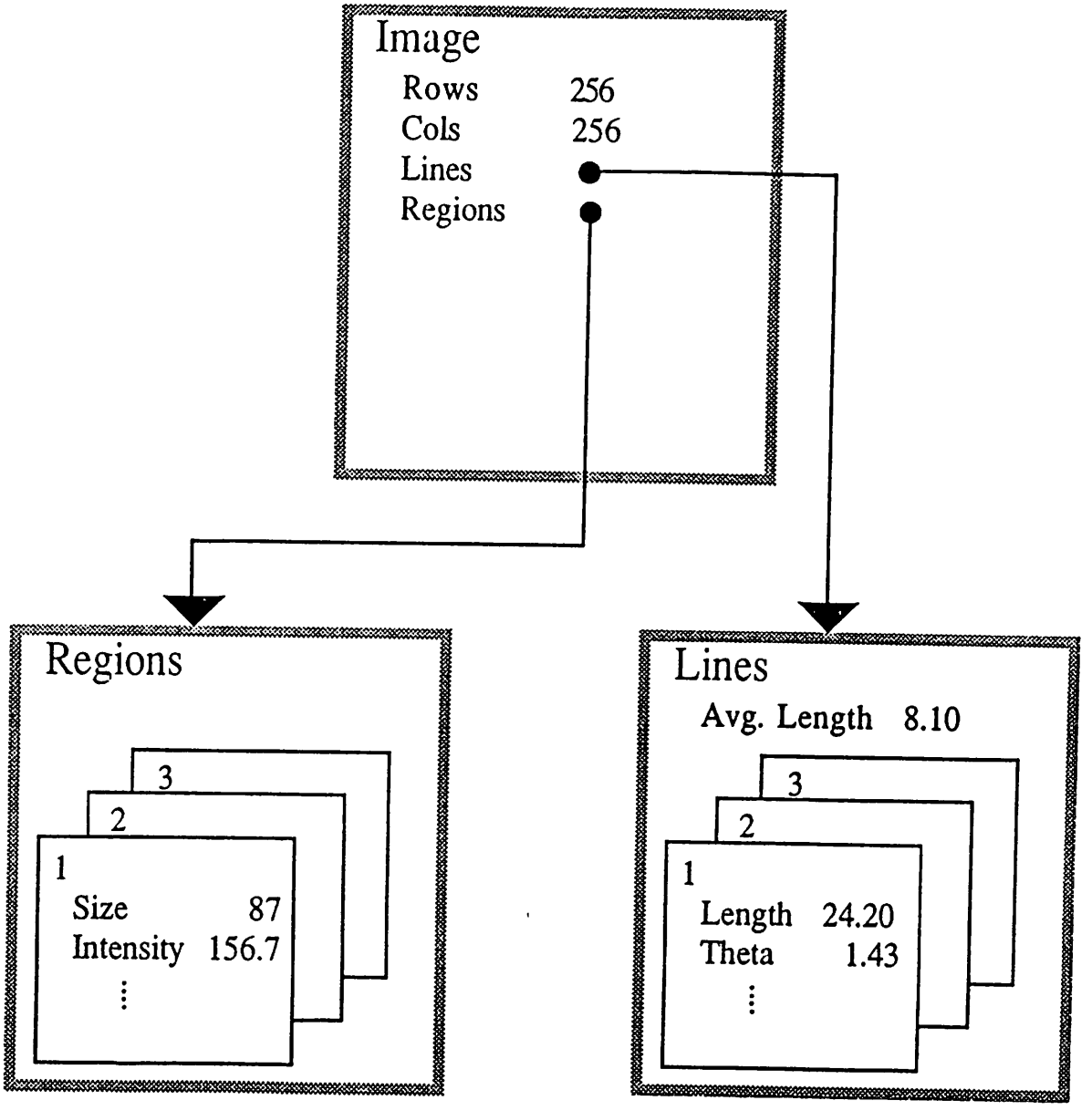
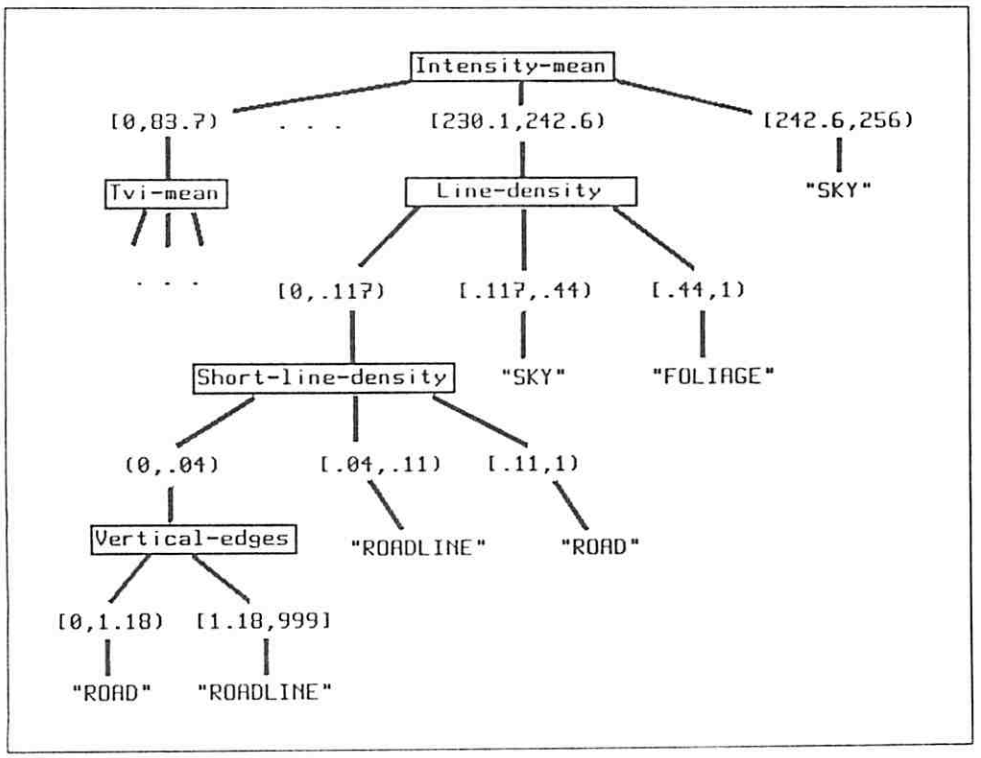
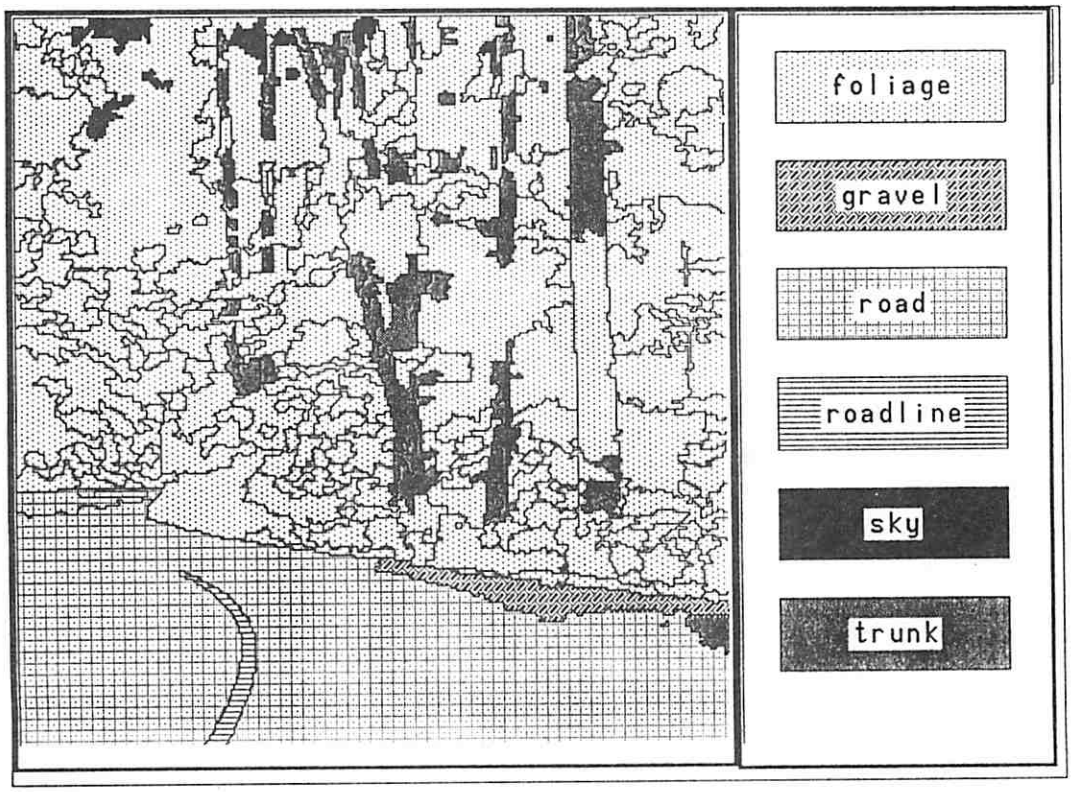


FIGURE 2 Brolio et al



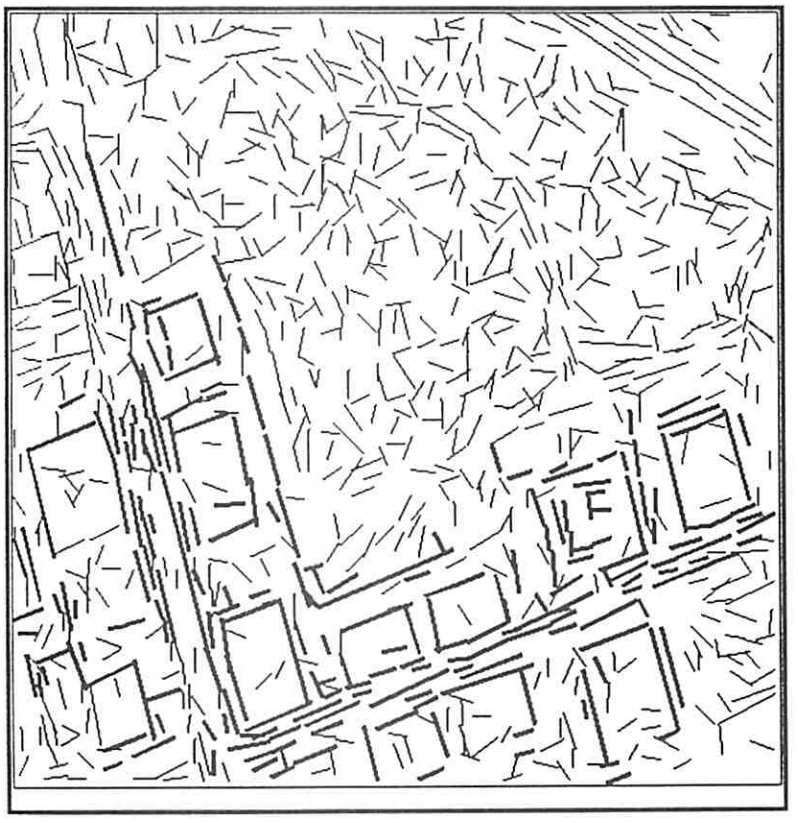
(a)



(b)

FIGURE 3 Brolio et al

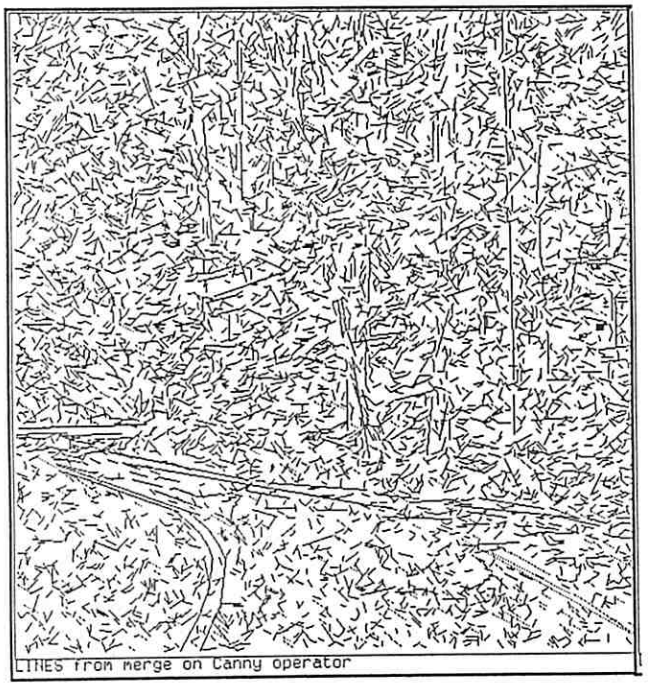
(a)



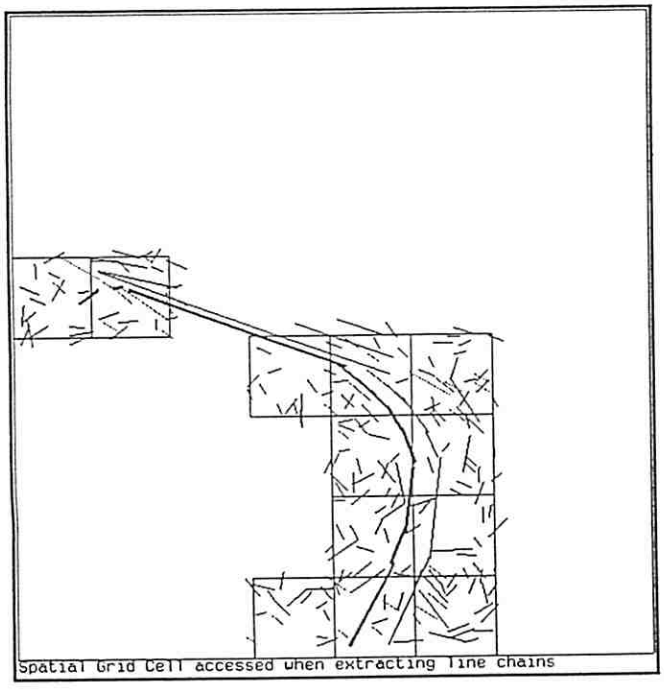
(b)

FIGURE 4 Brolio et al

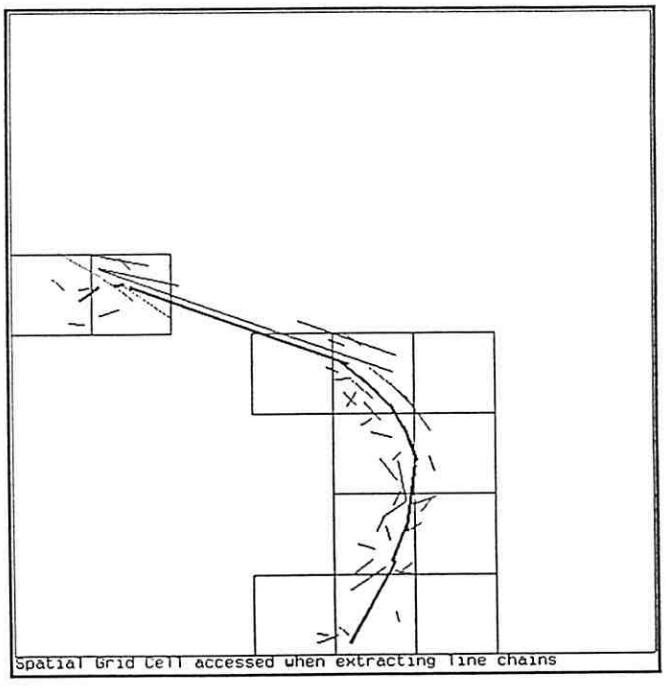
(a)



(b)



(c)



(d)

FIGURE 5 Brolio et al