ISR: A Database for Symbolic Processing in Computer Vision

John Brolio, Bruce A. Draper, J. Ross Beveridge, and Allen R. Hanson
University of Massachusetts at Amherst

Computer vision imposes unique requirements on the representation and manipulation of image data and knowledge. At a vision system's lowest level are sensors that represent an image with purely numeric image arrays, while at the highest level are semantic world models that provide the final interpretation of the scene. In between are thousands of intermediate descriptions, many of which must be repeatedly accessed and processed during interpretation. Traditional knowledge representation methods do not provide mechanisms to accomplish this effectively.

In this article, we describe a representation and management system for use at the intermediate (symbolic) level of vision. Based on database management methodology, the Intermediate Symbolic Representation (ISR) mediates access to intermediate-level vision data and forms an active interface to the higher-level inference processes that construct an image's interpretation. The system supports important types of data and operations and can be adapted to the changing needs of ongoing research. Furthermore, it provides a centralized data representation that supports integration of results from multiple avenues of research into the overall vision system.

A vision system generates thousands of intermediate-level descriptions. ISR combines database management and knowledge representation methods to allow effective access to these descriptions.

A computational paradigm for computer vision

The goal of computer vision is to construct automatically from a digital image a symbolic interpretation that describes the environment from which the image was captured. This description includes the identity and structure of objects and their spatial and temporal relationships. (We restrict our discussion to static images captured from a single sensor with no motion information.) The need to identify objects implies that vision systems must have access to a knowledge base containing descriptions of the objects or of generalized object classes. Consequently, a generally accepted paradigm for computer vision involves multiple levels of representation and abstraction, from image representation (purely numeric) to object or object-class models (highly abstracted) stored in the knowledge base.[1]

This paradigm underlies the Visions Image Understanding System, a research system being developed in the Computer Vision Laboratory at the University of Massachusetts at Amherst. Visions 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, and motion information, are extracted from the image data, typically without knowledge of the image content.

0018-9162/89/1200-0022$01.00 © 1989 IEEE

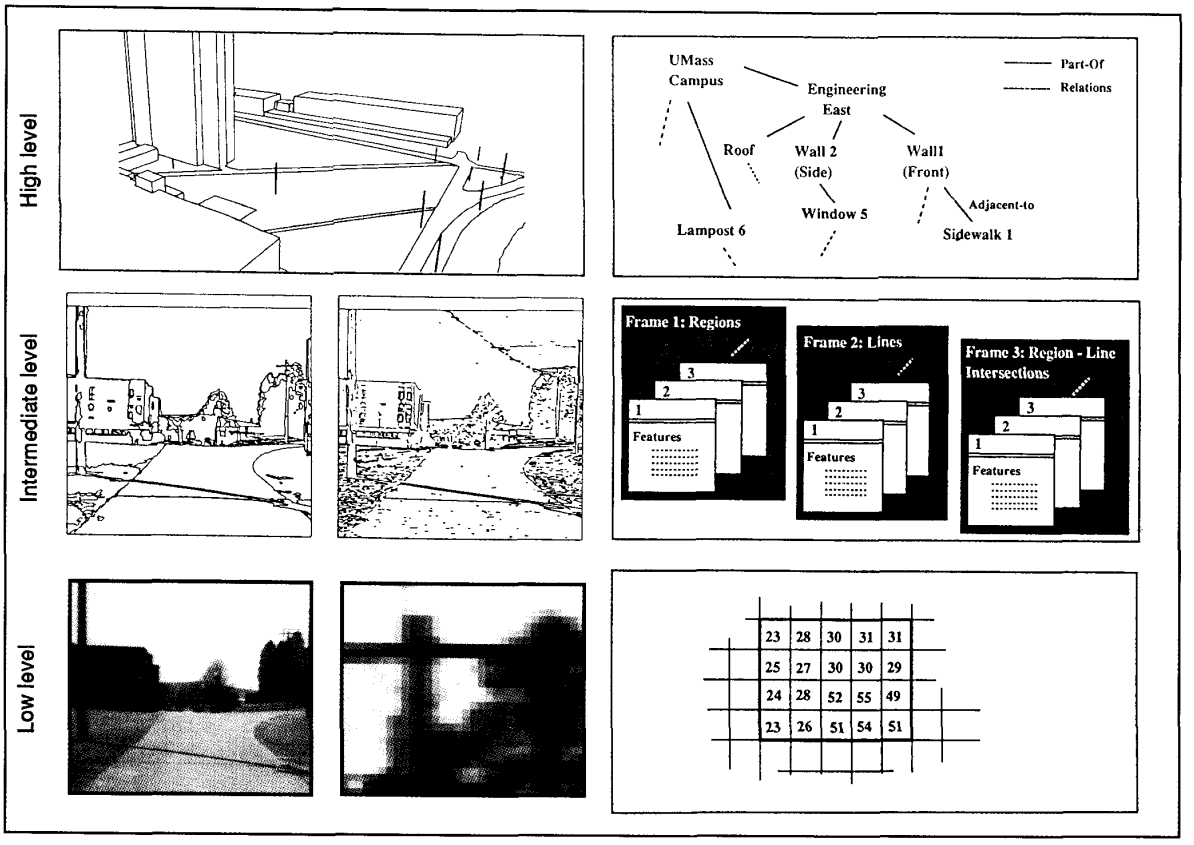


Figure 1. Stylized view of the three levels of representation in Visions. The left column contains representations of data at each level, and the right side shows the internal representations corresponding to these data. At the low level are the image data and an image blowup (left), with the internal representation consisting of numeric arrays (right). The intermediate level shows a region segmentation in which regions are denoted by their boundaries (far left) and a straight-lines representation (left); the internal representation is a symbolic token and frame representation (right). Finally, at the high level, a 3D wire-frame model (left) represents the image interpretation, while the internal representation takes the form of a symbolic network (right).

The system stores descriptions of these events at the intermediate level as named and typed symbolic entities called *tokens*. Each token contains attribute-value pairs that 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, or modifying existing tokens. Tokens also 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 that 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. For more detail on Visions, as well as image interpretation results on house and road scenes, see Draper et al.[2]

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 before or during the matching process. Much of the transformation in Visions occurs 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 that satisfy certain geometric properties and aggregates of regions and lines consistent with an object or object part in the knowledge base. We later discuss additional examples of aggregate tokens and the processes that create them. There is no practical limit to the variety or quantity of tokens 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.[3,4] This algorithm groups short lines to form longer ones in a recursive

Table 1. Size of the intermediate symbolic representation after low-level processing.

Image	No. of Regions	No. of Straight Lines** (Short/Other)	No. of Line-Region Intersections*	No. of Boundary Lines** (Short/Other)	No. of Boundary Line-Region Intersections*	Data size Including Token Features (in Mbytes)
Road1	187	3,981/911	1,517	1,756/206	724	1.6
Road2	307	4,062/963	1,868	2,811/212	751	2.0
Road5	427	4,019/1,051	2,736	3,733/379	1,415	2.4
Road10	311	4,209/996	2,111	3,246/242	853	2.1
Road16	222	3,744/733	1,112	1,305/187	775	1.5
Road25	356	4,364/949	2,354	3,730/285	980	2.3
House1	305	2,843/878	1,793	1,174/405	1,438	1.5
House7	168	3,522/900	1,823	2,239/242	711	1.6
House10	165	2,675/845	1,255	965/241	836	1.2

* Line-region intersections and features are not calculated for short lines.
** Short lines are less than five pixels long.
Note: All images have a resolution of 256×256 pixels.

cycle of linking, grouping, and replacement 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 might have 10,000 or more initial line tokens, the spatial relations at the heart of this and similar algorithms must be efficiently represented.

The amount of intermediate-level data involved in the interpretation of a single image is quite large. Table 1 lists nine typical images used in interpretation experiments and details the size and storage requirements for five types of symbolic image events (not including low-level image data). The event types are regions, straight lines, line-region intersections, lines lying along region boundaries, and region-boundary line intersections. The data in Table 1 represents only those image events and relationships between them that are extracted from the image before execution of any intermediate-level grouping process or schema interpretation strategy.

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.

(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 currently feasible except with highly optimized data management. 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 for basic functions. If the system is not in general use, data will not be shared, machine-readable data output from experiments will be effectively lost, and researchers will constantly reimplement the same data structures and manipulation procedures.

Therefore, we have designed ISR as a "reduced instruction set" for computer vision applications. ISR provides a set of primitive operations and representations that are easy to use and with which any intermediate-level applications can be built.

Primary requirements. In designing ISR, we have drawn from both database practice and knowledge representation technology, the former for efficiency and generality and the latter for flexibility. We also 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.

(1) Vision research requires spatial data types and retrieval methods not usually supported in standard database systems. An algorithm like the Boldt line extractor can make hundreds of thousands of spatial access queries on a single image. The algorithms we discuss in the sections entitled "Perceptual grouping" and "Spatial access" do less work in total, but the ratio of spatial queries to other database queries remains about the same. The spatial access functions that implement these retrievals must be implemented efficiently. ISR supplies methods for retrieving all tokens that intersect a given bounding rectangle or all tokens that intersect a region of arbitrary shape. This feature is the main focus of the ISR design; we are continuing to research optimal methods of spatial data representation and retrieval.

(2) Data structures should be standardized sufficiently to allow efficient sharing of data:

- Incremental saves and loads of partial data sets must be possible.
- Data must be saved in logical modules of reasonable size and scope.
- Loading selected records and selected fields of those records should be possible.

All of these requirements are fulfilled by database management systems (DBMSs); knowledge representation systems currently do not provide much capability or flexibility in this area.

(3) The researcher must be able to reconfigure the database dynamically, add fields or attributes to a record, or create a new data set with records or attributes from one or more existing data sets. Every researcher is free to alter the syntax, semantics, and facts in a database; keeping these changes private or making them available to the research group should be easy and not require storing large quantities of redundant data. In ISR, it is simple to create a data set, alter its definition, or copy, load, or save a data set's selection or projection. The system provides reasonable means of tracking the sources of various elements of a user's individual database.

(4) Like any research tool, the database application language must be as powerful and 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, we built a prototype of ISR and spent a year and a half analyzing information about its use, about spatial retrieval applications written or proposed for it, and about requests for changes or new features. We then integrated our findings into the current design.

(5) Extendability 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, requirements for data and database operations 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.

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

The traditional DBMS environment demands three separate levels of human interaction: 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 types of interaction. An individual researcher can 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 ISR's design.

Knowledge representation technology. Current knowledge representation technology has some positive qualities that are being incorporated into new database research.[5] However, it also has a number of deficiencies, some of which can be remedied by recourse to DBMS methods. The positive features are extendability, flexibility, and the capability for procedural attachment. We have borrowed these in the concept of *frames* (data types that can be redefined dynamically) and *demons* (procedures that can be executed when data field values are accessed).

Unfortunately, such common database utilities as sorting and indexing have been quite rare in knowledge representation systems, perhaps on the presumption that knowledge needs a great deal of hierarchical structure and very little linear structure. Vision and much other artificial intelligence research, however, requires full support of standard database operations and such standard data types as arrays, sets, integers, floats, and strings.

Data storage and retrieval are an afterthought in many systems, making saving and reloading data sets a cumbersome operation. Data sharing is extremely difficult because data values and data descriptions are stored as a unit.

Also, those inference procedures built into knowledge representation systems are generally wide of the mark. In an area as data-intensive as computer vision, 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, requiring 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 system already existed. Image DBMSs 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. Although engineering database systems require similar flexibility and extendability, they are even less appropriate because they offer nothing equivalent to low-level vision. What we required was not available: a system tailored to the needs of computer vision research and easily adaptable to different machines and languages.

There are other database systems for computer vision, most notably Carnegie Mellon's Codger[6] and SRI's Core Knowledge System.[7] Codger has evolved over the same time period as 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 as primitives many functions that 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 researchers in high-level vision might want a different methodology than the one supported. In the long run, such an all-encompassing approach could render the system cumbersome, making it less likely to be used by the researchers for whom it is intended.

The Core Knowledge System is a more recent design that seems to be intended as a knowledge representation system for 3D world modeling at a very high level. It devotes much attention to semantic issues that are less essential at the intermediate level of computer vision and hence pays less attention to efficient spatial retrieval and manipulation of 2D symbolic entities.

ISR seems to be unique as a small, efficient, and flexible DBMS focused on the needs of intermediate-level vision.

The ISR data management system

ISR's design was constrained by both the nature of symbolic image interpretation and the need to serve multiple researchers with varied interests. Conse-

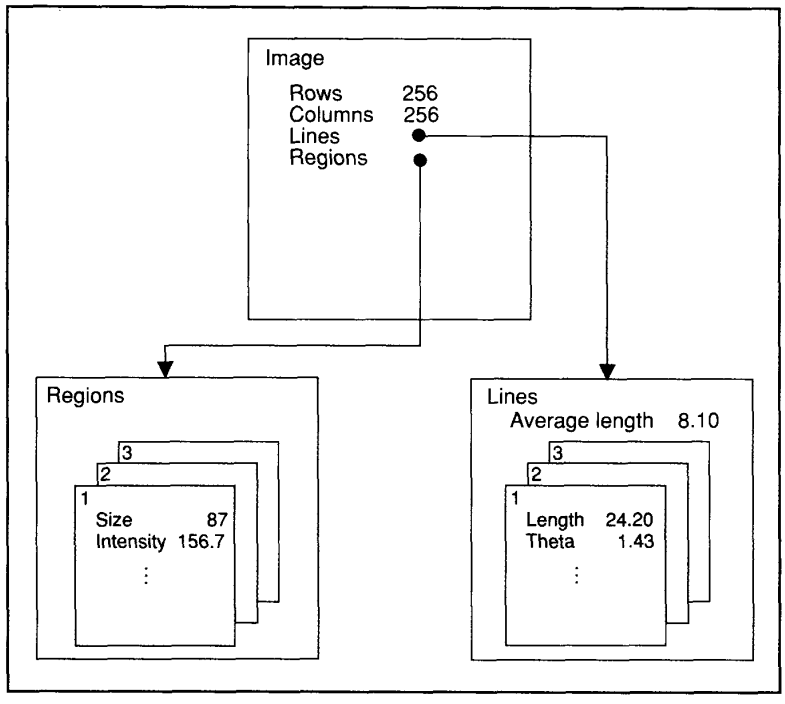


Figure 2. A frame and token hierarchy, depicting a set of line tokens and a set of region tokens, all extracted from the same image. Note that image features describe a particular image event, such as the length of a line, while frame features describe a class of events, such as the average length of a line.

quently, the system provides a small set of highly optimized primitive commands and representations that 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 can be placed in additional libraries.

Tokens and features. ISR's fundamental object is the token, which is similar to a record in a standard DBMS. A token can represent an image event (such as 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 hypothesized to belong to some object). The data fields of an ISR token, called *features*, describe attributes of the referenced event. Features resemble frame slots in a knowledge representation system, in that they have attached procedures (demons) that can be activated whenever a value is requested or modified or when a value is needed that has not yet been computed. ISR tokens for similar image events are grouped into token sets, which allow operations over similar data. For example, a token set consisting of all the lines for one image would allow such operations as displaying every line in the set or computing the contrast across each line.

Frames. Each token set is embedded in a descriptor object called a frame. The relationship between a frame and its token set is similar to the class/instance relationship 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.

Like tokens, frames are first-class data objects with features. Frames are also linked into hierarchies that denote relationships between associated token sets. 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). In addition, the frame provides a modifiable description of its token set. For example, a frame feature can contain statistics on token set 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 might not fit another, the user can specify the frame hierarchy.

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

Since feature storage is allocated at runtime as needed, the user can create a virtual feature whose value is computed as needed and never stored. This is useful for defining data aggregations that are constructed on demand from primitive features. For example, assume that the endpoints of a straight line are stored as four features, x_0, y_0, x_1, and y_1. If a user program needs to access each endpoint as a vector, the user can create a virtual feature called endpoint. 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 up the vector 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-/world-centered coordinates.

Handles and subsets. One of our fundamental requirements for ISR was that it express relations between tokens, especially associative relations. ISR can express simple pair relations through token features called *handles*. A handle is a reference to a token or frame in the database. Associative relations are represented by token subsets, which specify some or all of the tokens in a token set. For example, a token subset can denote the set of all regions whose average intensity exceeds 50 grey levels or all lines that are at least five pixels long. We can view ISR functions

that create and manipulate token subsets as a software analogue of content-addressable or associative memory. Token subsets can be

(1) selected by numeric feature ranges or properties of pixel maps;
(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 more than five pixels long. A special class of token subset, called a *sort*, represents an ordering over the token subset's elements. Token subsets and sorts are both handles and can be stored on any token or frame feature of that type.

ISR in use

Classification. One of the oldest and most studied problems in computer vision is the classification of image regions by feature values.[8] 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 that takes advantage of lazy evaluation is based on *decision trees* (also called regression trees[9]).

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 is exited 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 partly because of their efficiency. Unlike a Bayes classifier, a decision tree only requires that a few of a region's features be computed, specifically, the features on the path from the root node to the eventual leaf node. To take advantage of this, ISR supports lazy evaluation. When 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 help avoid computing unnecessary feature values.

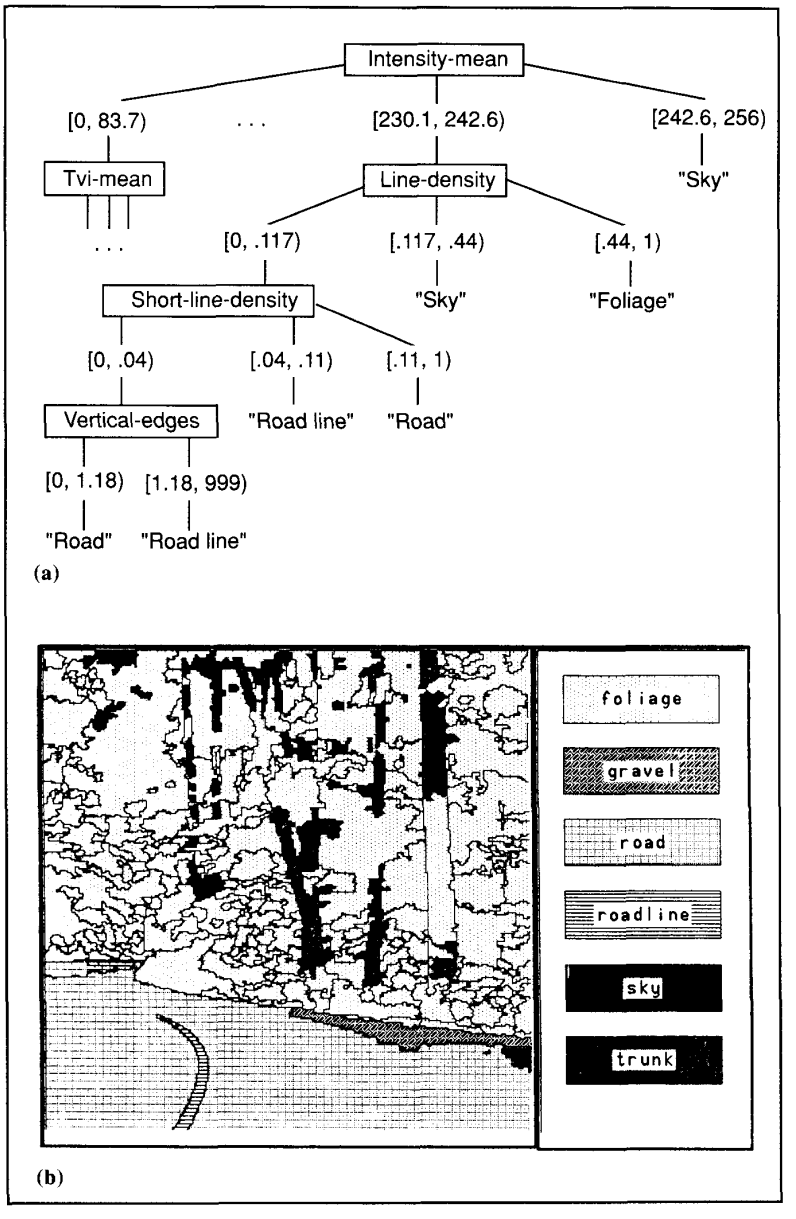


Figure 3. Region classification: (a) a portion of a decision tree that classifies regions as objects on the basis; (b) the result of classifying regions based on the tree in (a). The original image is shown in Figure 5a.

Perceptual grouping. Grouping related tokens into aggregate structures (perceptual grouping) is a common intermediate vision task. One strategy groups tokens by the transitive 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 that represents a new image event and can possess properties not

Figure 4. Results from rectilinear line grouping: (a) the original aerial image; (b) one of the rectilinear line groups (in heavy lines) found by the grouping algorithm, superimposed over the complete set of intermediate-level straight lines that constitute the input to the algorithm.

possessed by any of its components.

The Rectilinear Line Grouping System[10] is a perceptual organization system in 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 token subset on the line token. The RLGS repeats this process, 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 group. The group can either be returned to the user as a token subset or be 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.

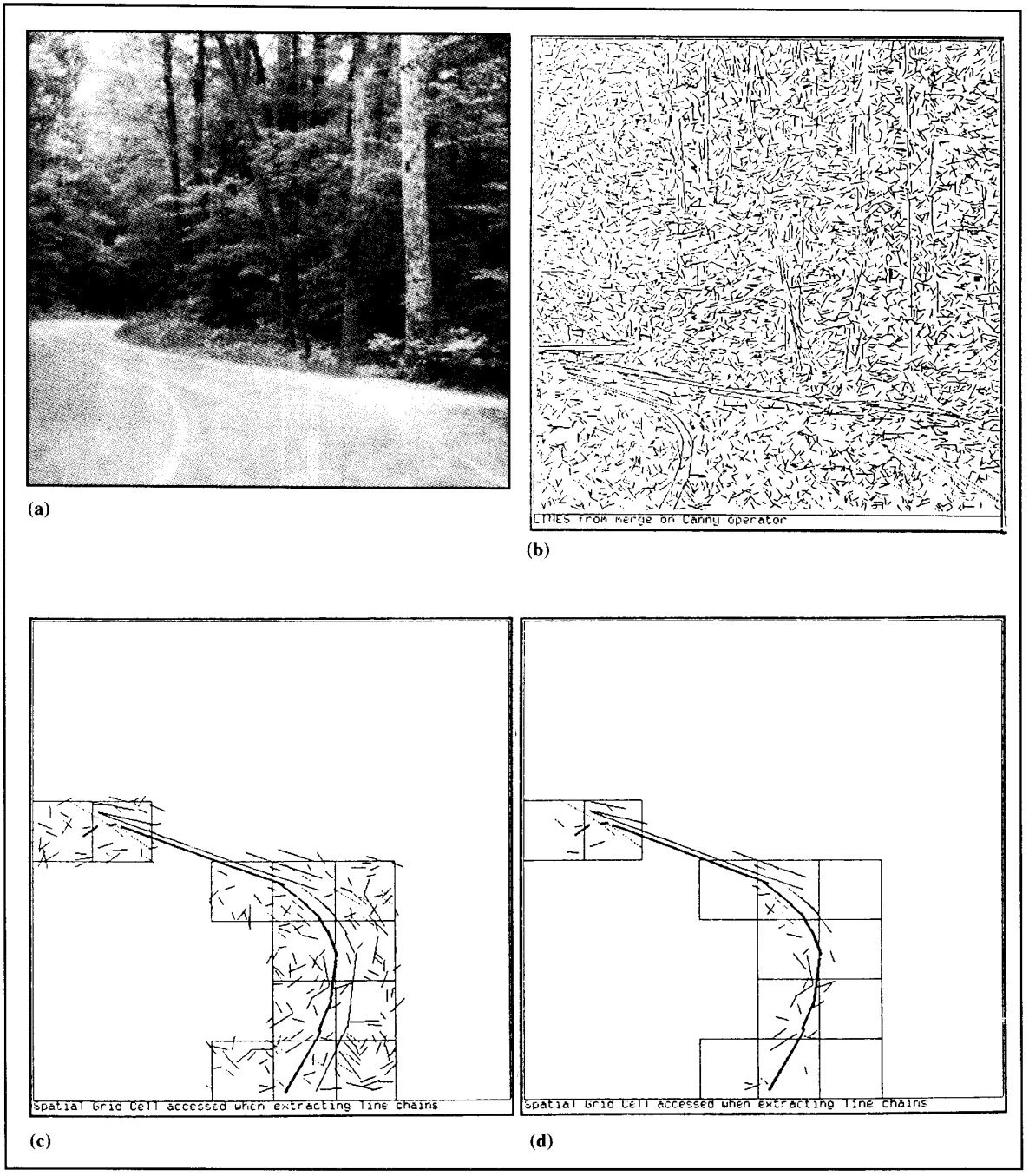
Spatial access. Spatial proximity is an important consideration in accessing image information. We often want to access just those tokens lying on or near a particular point in the image. To this end, grids are often imposed on the 2D image, dividing it into rectangular cells called *bigcells*. If lines, regions, or other image events are stored according to the bigcells they intersect, it becomes relatively easy to retrieve a token's neighbors by accessing only those bigcells that lie within the radius of interest.

We have implemented bigcells in ISR by creating a token set 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) that spatially intersect that cell. We have written functions 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.[2] Figure 5a is a photograph of a typical road scene, and Figure 5b is the output of a low-level algorithm for identifying lines applied to the lower left quadrant of the photo image. The goal is to identify the boundary of a road line 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 that 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 (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 that is a good candidate for a boundary line. That line is extended by joining it end-to-end with other lines that meet it near its endpoints. The algorithm selects candidates for extension by retrieving any lines within a small radius of the endpoints of the current line-chain. Figure 5c shows (along with the final successful line-chain interpretation) the grid cells accessed during the search and the candidate lines retrieved. Figure 5d 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.



(a) (b) (c) (d)

Figure 5. Bigcell spatial access during image interpretation: (a) the original image; (b) the set of straight-line tokens extracted from the image; (c) the grid bigcells accessed during the search and the candidate lines retrieved; (d) those candidates within a small distance of the current line-chain endpoints.

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 token sets on the corresponding cells. Thus we can use the same primitives that implement spatial access to implement histogram-based algorithms and generalized Hough transforms.

Our experience with intermediate-level computer vision has taught us three lessons:

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

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, ISR provides special spatial representations and access routines not traditionally found in either of these two fields.

Versions of ISR have been operating in the Computer Vision Laboratory at the University of Massachusetts 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 ISR has recently been embedded in a commercial system.

Our major conclusion is that neither traditional DBMSs nor knowledge representation languages support the primitive spatial data structuring and access functions required at the intermediate level of vision. ISR adds spatial primitives to the capabilities of database management and knowledge representation systems to provide a simple yet effective database for intermediate-level vision. ■

Acknowledgments

We are grateful to Bob Collins for feedback during the design of ISR, for the development of the decision tree application, and for producing Figure 3. We also gratefully acknowledge the help of Joey Griffith, Bob Heller, Ric Southwick, and Jim Burrill in designing and implementing various versions of ISR. In addition, we would like to thank the many members of the University of Massachusetts Computer Vision Laboratory who have used ISR and given us invaluable feedback. This research has been supported in part by DARPA under contract number F30602-87-C-0140 and NSF DCR-8500332.

References

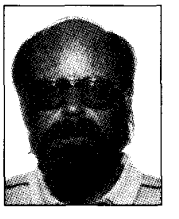
1. A. Rosenfeld, "Image Analysis: Problems, Progress and Prospects," *Pattern Recognition*, Vol. 17, No. 1, 1984, pp. 3-12. Also appears in *Readings in Computer Vision*, M.A. Fischler and O. Firschein, eds., Morgan-Kaufman, Los Altos, Calif., 1987, pp. 3-12.
2. B.A. Draper et al., "The Schema System," *Int'l J. Computer Vision*, Vol. 2, 1989, pp. 209-250.
3. M. Boldt, R. Weiss, and E. Riseman, "Token-Based Extraction of Straight Lines," to be published in *IEEE Systems, Man, and Cybernetics*.
4. R. Weiss and M. Boldt, "Geometric Grouping Applied to Straight Lines," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1986, CS Press, Los Alamitos, Calif., Order No. 721, pp. 489-495.
5. M.R. Stonebraker, "Object Management in Postgres Using Procedures," *Proc. Int'l Workshop Object-Oriented Database Systems*, Sept. 1986, CS Press, Los Alamitos, Calif., Order No. 734 (microfiche only), pp. 66-72.
6. S.A. Shafer, A. Stentz, and C.E. Thorpe, "An Architecture for Sensor Fusion in a Mobile Robot," *Proc. IEEE Int'l Conf. Robotics and Automation*, 1986, CS Press, Los Alamitos, Calif., Order No. 695, pp. 2,002-2,011.
7. M.A. Fischler and R.C. Bolles, "Image Understanding Research at SRI International," *Proc. DARPA Image Understanding Workshop*, 1989, pp. 21-31.
8. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.
9. L. Breiman et al., *Classification and Regression Trees*, Wadsworth, Inc., Belmont, Calif., 1984.
10. G. Reynolds and J.R. Beveridge, "Searching for Geometric Structure in Images of Natural Scenes," *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 257-271.

John Brolio is a staff member of the Generic Blackboard Development Group at the University of Massachusetts. He received his MS degree in computer and information science from the university in 1987 in the area of computational linguistics. His other area of interest is large-scale data and knowledge representation for AI systems.

Readers can contact the authors at the Computer and Information Science Department, Lederle Graduate Research Center, University of Massachusetts at Amherst, Amherst, MA 01003.

Bruce A. Draper is a doctoral student in computer science at the University of Massachusetts. His interests are in computer vision and artificial intelligence. Draper is a student member of IEEE and AAAI. He received his BS degree in computer science from Yale in 1984 and his MS degree from the University of Massachusetts in 1987.

J. Ross Beveridge is a doctoral student in computer science at the University of Massachusetts at Amherst. His interests include computer vision, robotics, and artificial intelligence. He is a student member of IEEE and AAAI. He received his BS degree in applied mechanics and engineering science at the University of California, San Diego, and his MS degree in computer science at the University of Massachusetts.



Allen R. Hanson is professor and associate director of the Computer Vision Laboratory at the University of Massachusetts. For the past 15 years, his research efforts have been in artificial intelligence, computer vision and image understanding, and pattern recognition.

Hanson is coeditor of *Computer Vision Systems* (Academic Press, 1978) and *Vision, Brain, and Cooperative Computation* (MIT Press, 1987), coauthor of *Fundamentals of the Computing Sciences* (Prentice-Hall, 1978), and an editorial board member of several journals. He is a founder of Amerinex Artificial Intelligence Corporation and VI Corporation, both located in Amherst, Mass.

Hanson received the BS degree from Clarkson College of Technology, Potsdam, New York, in 1964, and the MS and PhD degrees from Cornell University, Ithaca, New York, in 1966 and 1969, respectively, all in electrical engineering.