Image Interpretation by Distributed Cooperative Processes

Bruce A. Draper, John Brolio, Robert T. Collins, Allen R. Hanson, Edward M. Riseman

Department of Computer and Information Science
University of Massachusetts
Amherst, Mass., 01003.
(413) 545-0528

Special-purpose vision systems have shown considerable success within limited task domains, but to date there are no general-purpose systems working effectively across a variety of task domains. The VISIONS Schema System[1] provides a framework for building a general interpretation system as a distributed network of many small special-purpose interpretation systems. Each schema is an "expert" at recognizing one type of object. Schema instances run concurrently, communicating asynchronously through a global blackboard, each trying to establish the presence of its particular object by invoking general purpose knowledge sources. In this way schema instances cooperate and compete to identify and locate the significant objects present in the scene. This paper discusses the problems of knowledge representation in a distributed AI environment, and the Schema System's approach to those problems. A series of interpretation experiments have been performed on nine images from two natural scene domains; results from three images will be presented here. Interested readers are referred to [6] for a full description of the system and the experiments.

1. Knowledge-Based Vision Systems

Special-purpose vision systems (e.g. [21,22,18,2]) have succeeded because they are uniquely able to define, structure, and apply knowledge relevant to their task domain. Knowledge in vision includes domain-independent knowledge about occlusion, perspective, physical support, etc., as well as domain-dependent object knowledge about attributes and relations, and object-specific control knowledge for recognizing objects in scenes. Systems working in restricted domains can bring very specific object and control knowledge to bear on their task. Very small domains admit the possibility of a complete world model, offering constraints which permit sophisticated inferences with comparatively little computation. General-purpose systems, using only general structural knowledge and inference machinery, are often unable to solve non-trivial problems [3]. "Such problems are only solved by the use of a large, domain-specific knowledge base. It has become almost an axiom of artificial intelligence that powerful problem solving in any realistic domain requires a large amount of knowledge" [15].

[1]This work has been supported by the Air Force Office of Scientific Research under grant number AFOSR-86-0021, by the Defense Advanced Research Projects Agency under contract number F30602-87-C-0140, and by the National Science Foundation under grant number DCR-8318776.

The UMass Schema System represents an attempt to build a general-purpose vision system out of many special-purpose ones. The Schema System is a knowledge-based, high-level component of the VISIONS [10] image understanding system. The system goal is to interpret static, color images by identifying and locating the significant objects in the scene and identifying relevant object relationships. Knowledge bases currently exist for two natural scene domains: house scenes and road scenes.

Our primary design philosophy is that both knowledge and computation should be partitioned at a coarse-grained semantic level. Each schema is specialized to identify one particular class of object. A schema instance is invoked for each object instance hypothesized to be in the scene. These schema instances run as independent concurrent processes, communicating asynchronously through a global blackboard when necessary. Depending on the hardware, these processes can be distributed among available processors. Each schema instance directs the application of general-purpose knowledge sources to gather support for its object hypothesis. The goal is to foster cooperation and competition among schema instances, resulting in a set of object hypotheses which are both semantically and spatially compatible.

The experience of other researchers, as well as our own, has shown that the choice of a *knowledge representation* can greatly affect the ease with which a system is built, and its efficiency when running. Most knowledge-based systems encode information in the form of rules, frames, blackboard knowledge sources, or logic-based languages. These representations are epistemologically equivalent; the differences are in efficiency, documentability, and ease of implementation/extension. Nagao [18], developed a production system variant in which the rules were actually complex vision subsystems. Ohta [19] used a more traditional production system, supplemented with certain non-modular mechanisms for control knowledge. McKeown's SPAM [16] has a production system knowledge base of over 500 rules, organized into five processing phases, for interpreting aerial images of airports. Glicksman [8] used a frame-based knowledge representation in his human-aided interpretation system, and Hwang [12] controlled a frame system by means of a blackboard-based scheduler. Shafer *et al.* [20] have developed a system for real-time navigation whose global organization is similar to the one described here in that they use a set of visual "knowledge sources", each running continuously on its own processor. At UMass, Hanson and Riseman [9] described a frame-like "schema system" for computer vision, which was inspired in part by the work of Arbib [1] and Minsky [17]. Later, Weymouth [24] demonstrated initial success at interpreting natural images with schemas running in a simulated distributed environment. The current UMass Schema System represents the continuing evolution of these ideas.

2. The Schema System

The Schema System partitions both knowledge and computation in terms of natural object classes for a given domain (e.g. SKY, ROAD, HOUSE). Each class of objects and object parts has a corresponding schema which stores all object and control knowledge specific to that class. To identify an instance of the object class in an image, a *schema instance* is created. A schema instance is an executable copy of the schema which runs as a separate process with its own local state. The system's initial expectations about the world are represented by one or more "seed" schema instances which are active at the beginning of an interpretation (e.g. ROAD-SCENE, HOUSE-SCENE, or the more general OUTDOOR-SCENE). As these instances predict the existence of other objects, they invoke the associated schemas, which in turn may invoke still more schemas. From the instances thus spawned, the "successful" instances establish a set of object hypotheses which constitute the interpretation of the image. At any point during processing the current partial interpretation is made up of those hypotheses with the strongest measure of support from the data and form consistency relations with other hypotheses.

2.1 Communication

While each schema can be viewed as a local expert subsystem for recognizing its associated object, schema instances must be able to communicate to arrive at a consistent interpretation. A schema needs information about the presence or absence of subparts, contextually related objects, possible occluding or shadowing objects, and hypotheses which offer alternative explanations. Communication in the Schema System is achieved through a global blackboard. The global blackboard allows schema instances to publish their contributions to the incrementally developing interpretation and to access the public contributions of other schemas. The benefits from the blackboard's decoupled communication are in terms of flexibility and modularity. The posting schema does not need to know which (if any) schemas will look at its hypothesis, nor do the posting and reading schemas have to be simultaneously active.

In the current system, the global blackboard is divided into sections, one section for each object class. Any schema can read from or write to any section of the blackboard – the division is for retrieval efficiency only. Dividing the blackboard into sections gives some assurance that a schema will not have to search through a large number of irrelevant messages. Sectioning the blackboard by object class is in contrast with the processing level partitioning common in other blackboard and rule systems (e.g. [7],[16]).

All messages written to the global blackboard adhere to a small number of rigid formats which are consistent throughout the system. The most common message type is the *object hypothesis*, which includes the object class, part/subpart links to other object hypotheses, a list of image areas claimed or explained by this hypothesis, and a confidence value. Confidence values are currently chosen to lie along a coarse, five-point ordinal scale ranging from *no-evidence*, the lowest value, through *slim-evidence*, *partial-support*, *belief*, and finally *strong-belief*. This scale was chosen arbitrarily to represent a small number of degrees of belief to distinguish the different amounts and quality of evidence supporting a hypothesis. Other public message formats currently in use are 1) a spatial index from image area to the object hypotheses claiming it, and 2) a conflict message, posted whenever two incompatible hypotheses overlap spatially.

In addition to public object hypotheses, a schema instance also maintains internal hypotheses about possible instances of its object class. Internal hypotheses contain a record of the evidence gathered to support or deny the presence of the object instance in the image. The schema stores this internal knowledge on its own local blackboard. Representations on the local blackboard are private and local to the owning schema. This permits the schema designer the freedom of any appropriate knowledge representation and control style, while at the same time protecting the Schema System from an overabundance of public message formats. A schema maintains an internal representation of its hypotheses even after it has published them, so that if relevant new information is received the schema can recompute its confidence in the hypothesis due to new support, or perform further analysis to strengthen a hypothesis weakened by a loss of supporting evidence. This provides a limited and local form of truth maintenance.

2.2 Schema Construction

Although each schema is viewed as a special-purpose vision subsystem, building every schema from scratch would be an unmanageable task. The VISIONS environment provides a set of building blocks in the form of Knowledge Sources (KSs). KSs are general-purpose routines called by the schemas. While they may contain knowledge associated with their particular task, they do not contain object-specific knowledge, and thus are useful over a variety of objects and domains. Sometimes a KS is a complex subsystem that is itself a topic of research, as in the case of the Graph-Based Matcher (GBM), or the Initial Hypothesis System (IHS) (see [5] where they are called OVARC and Rulesys, respectively). It is important that the amount of computational work a KS performs be controlled by the calling schema. For complex KSs this means that the major control factors must be parameterized. In addition to the KSs, VISIONS provides a database of predefined object descriptions and image event representations ([5], the ISR database).

Each schema models the likely *appearance* of an object in the image. The schema encodes appearance information in terms of extractable image tokens, attributes, and relations, and in particular, which combinations of these imply the presence of the object in the scene. This may lead to several distinct characterizations of the object, since appearance changes with respect to different viewpoints and scales. Once a set of likely object appearances are identified, a schema is constructed by assembling the appropriate knowledge sources into a control strategy that specifies when to apply them and how to interpret their results. More precisely, the schema designer 1) defines a *support space*, the possible sources of positive or negative evidence for the presence of an instance of the object class, 2) describes a set of paths and branches to traverse that space, taking into consideration the efficiency of the knowledge sources being applied and the likely quality and importance of the evidence returned, and 3) supplies a function to translate internal evidence from the knowledge sources into a confidence value for the object's presence in the scene.

The current road schema, for example, assumes a viewpoint of a driver looking down a paved road. Roads have no particular linear structure (especially in New England), therefore we rely on the region segmentation to provide candidate road tokens. As much of the foreground should be covered by the road, we expect the segmentation to produce relatively large road regions having a grayish (macadam surface) color and relatively low texture. Since the road may be mottled by shadows, we also expect that parts of the road area will be fragmented into smaller regions, particularly where the image of the road thins down as the road recedes into the distance. We also know about several objects which frequently co-occur with paved roads, namely painted roadlines, gravelly shoulder areas, telephone poles, and traffic signs.

Schemas represent accumulated evidence as symbolic *endorsements.* Some endorsements are produced by invoking knowledge sources on descriptions of the image, others are gathered by monitoring the global blackboard for hypotheses posted by related schemas. The endorsements recognized by the road schema are

```
color attributes (one of the following)
    :correct-road-color
    :neutral-road-color
    :wrong-road-color
texture attributes (one of the following)
    :correct-road-texture
    :neutral-road-texture
    :wrong-road-texture
related object relationships (any subset of the following)
    :favorable-roadline-relationship
    :near-shoulder
    :near-warning-sign
    :near-stop-sign
    :near-phonepole
road extension relationships (any subset of the following)
    :adjacent-to-road
    :above-road
```

The set of all possible evidence combinations comprises the schema's support space. There are thus $(3C1)(3C1)2^7 = 1152$ states in the road schema support space.

Each object schema maintains a confidence function which maps sets of support states (or sets of sets of endorsements) into one of the five levels of the confidence scale. For instance, two sets of states mapped by the road confidence function into a confidence level of *belief* are

```
;;correct color and texture, and supported
;;by at least one related object
(and :correct-road-color
     :correct-road-texture
     (or :adjacent-to-road
         :favorable-roadline-relationship
         :near-shoulder
         :near-warning-sign
         :near-stop-sign
         :near-phonepole))
```

and

```
;;adjacent to and above (in the 2D image) a believed
;;road region, and having the correct relationship (in
;;terms of orientation) with a believed roadline
(and :adjacent-to-road
     :above-road
     :favorable-roadline-relationship)
```

Other sets of states map to lower levels of confidence.

The goal of a schema is to collect the evidence necessary to post a hypothesis of confidence level *belief* or higher. This process can be viewed as tracking a hypothesis through the support space. Each test applied by the schema adds new endorsements to a hypothesis, moving that hypothesis to a new support state and possibly changing its confidence level. The order in which a schema applies its tests reflects a path through the support space. Some paths have higher computational costs than others. A schemas control strategy determines which paths to take, based on the schema designer's intuitions about knowledge source complexity and reliability. Among the several control heuristics employed by the road schema, two are

Compare region attributes against apriori color and texture expectations first, since this is a relatively cheap operation. Regions having the wrong color and texture can be excluded from future consideration.

Only after a portion of the road hypothesis has reached *belief* should the search for road extensions occur.

3. Schema System Issues

3.1 Knowledge Representation

3.1.1 Frames, Rules, and Blackboard Knowledge Sources

Knowledge in AI systems has typically been represented in frames, rules, blackboard knowledge sources, or logic-based declarative languages.[2] In this section we compare and contrast the representation of knowledge in the Schema System with these alternatives, and argue that the Schema System architecture can be viewed as the natural evolution of a blackboard architecture in a distributed environment.

Frames and *frame systems* have been popular in AI. As data structures, frames offer the benefits of record structures, slot access procedures (*demons* or *active values*), and value inheritance. In a pure *frame system*, however, the frame is a control mechanism as well as a data representation. The control paradigm of a frame interpreter is essentially two-phase; a structure-matching phase and a forward chaining phase. In the structure-matching or slot-filling phase, a candidate frame instance is chosen by some focus-of-attention criterion, such as "nearest to completion," and the database is searched in an attempt to fill the remaining slots in that frame instance. Where there are constraints on the slots which reference other slot values or other frame instances in the system, structure matching devolves into a form of subgraph isomorphism, with the accompanying potential for combinatorial explosion. Great care must be taken in defining constraints for a frame so that the search and match can be effective.

[2]For the sake of the arguments in this paper, we will view logic-based approaches as being equivalent to backward-chaining rule systems.

The forward chaining phase is initiated by the attached *if-added* demon when a slot is filled. It is more efficient than blind forward chaining schemes in that a demon is not associated with a slot unless its action has been deemed meaningful, thereby minimizing irrelevant inferences. In addition, the representation of the problem domain in a frame system appears reasonably clean, since much of the procedural complexity can be hidden in the attachments. This appearance can be extremely deceptive, however, when the attachments carry much of the burden of interpretation, precisely because the procedural behavior is concealed, and not temporally coordinated. In the monolithic frame system, the designer possesses a snapshot of the network of data relations, but essential questions about ordering, branching and looping behavior are extremely difficult to answer. This makes frame representations ideal for small, tightly integrated parts of a domain, where much of the knowledge can be captured in a restricted and relatively homogeneous implementation. However, imposing a frame representation blindly on *all* aspects of a computational domain can result in extreme inefficiency and opacity.

In *rule systems*, knowledge is encapsulated in if-then rules, or condition-action pairs, which interact with the body of data in working memory. Because of the large number of rules needed in a real-world task domain, additional control has been found indispensible and has usually been added by structuring the rule-base into classes [19] or phases [16], or by implementing additional rule sets which make control decisions. This last approach defeats modularity and makes system modification difficult. Rule classes or phases represent control information explicitly, but the control decision to test a rule still resides in at least two separate locations (the rule's class declaration and any rules which invoke the class) so that a change to either location may have unforeseen effects on the computation. In addition, there is a tradeoff between class size and the number of classes. At one extreme is a system with a few clearly marked classes or phases, each of which contain a large number of fine-grained rules (e.g. [16]); at the other extreme there are a large number of classes, each having a few complex, coarse-grained rules (e.g. [18]). This latter solution naturally leads to a blackboard style of control, where *events* on the blackboard signal when a rule's condition part should be matched against working memory.

A *blackboard knowledge source* (BBKS) is the equivalent of a rule[3]. In addition to the condition and action components, each BBKS also declares a set of triggering blackboard events, one or more of which must occur before the BBKS's condition predicate can be tested. The declaration of triggering events gives the BBKS a measure of the state of the system which is more fine-grained than classes or phases, but more economical than exhaustively testing each rule predicate.

We have adapted much of the structure of the basic blackboard system, with a few significant adjustments. A great deal of work on control and scheduling in blackboard systems [11],[14] uses centralized control, so that a BBKS must have some way of indicating to the central controller why it is running, what resources it will consume, and what it might produce in the way of output. There may be a large number of parameters that a BBKS must "tweak" in an effort to influence the scheduler [11]. Where a group of BBKSs form a natural processing sequence or tree, it may thus become quite difficult for the system user to manipulate the scheduler in order to achieve the necessary effect. Furthermore, it is possible that a large investment in centralized control might prove fatal in a distributed system.

[3]Blackboard system knowledge sources (BBKSs) fall somewhere between our schemas and our knowledge sources (KSs) in both power and grain size. The two types of knowledge source (KS and BBKS) discussed here should not be confused. Our knowledge source is a function or set of functions which do not conceptually interact with the blackboard at all.

The lack of intra-BBKS continuity is perhaps the most annoying problem for the AI researcher. Large scale perceptual systems do not always break down comfortably into chunks the size of the "READ RUN POST DIE" cycle of the BBKS instantiation. This problem was recognized as early as Hearsay-II, where terminating knowledge sources were allowed to dump their internal state onto the blackboard so that they could be resurrected and resumed at a later time [7].

3.1.2 Schemas

The Schema System gives a schema instance more intelligence, more autonomy and more continuity than a BBKS, thereby reducing the centralized control duties of the blackboard and the scheduler. The state of the interpretation is on the blackboard; if the available processing resources can be published in a similar manner, then there is no intrinsic reason why a schema cannot determine its own priority. The benefits are greater efficiency in a distributed system (due to the reduction in control communication overhead and the elimination of a potential bottleneck) and reduced burden on the schema designer. Furthermore, schemas can read the blackboard as often as necessary and suspend processing for long or short periods with no difficulty. The schema designer can think of a schema simply as a program with a set of concurrent interpretation strategies, an internal measure of success, and a means of translating its internal results into a uniform public hypothesis. It should be noted that other vision researchers using blackboards have also seen the need for continuous concurrent processes that maintain their own state [20].

3.2 A Distributable System

3.2.1 Distributed Control

We mentioned the potential control bottleneck embodied in the centralized blackboard agenda and scheduler. We hope to reduce this bottleneck by distributing scheduling decisions among the schemas themselves. The issue of when the cost of control processing outweighs the benefits is still an open question. In general, that process should run which incurs the least computational cost while providing the greatest contribution. Although cost is not too difficult to calculate, a potential contribution can only be assessed in the light of the goals of the computation.

In current blackboard systems [11], the solution is for the control knowledge sources to test at one time or another all of the other scheduled BBKSs in the system to determine which ones should be run and in which order. (The inefficiency of this testing frenzy can be mitigated by indexing [4].) We feel it will prove more economical, more modular and more effective to publish the goal information which the control KSs in a blackboard would use, and let the schemas themselves decide whether or not to run. In the worst case, control in the Schema System should have a cost/benefit behavior identical to that of the traditional blackboard, but generally the advantage should lie with the Schema System. Schemas suspend processing while waiting for *a particular piece* of control information, so that the processing queue

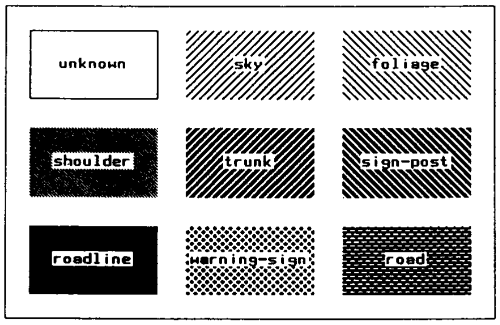

Table 1: Road Scene Shade to Object Encoding

(agenda) should be significantly smaller. Thus, control processing should benefit from the same modularity and opportunistic focus behavior as does the domain processing.

3.2.2 Distributed Communication

We have attempted to reduce the blackboard *information* bottleneck in two ways: 1) each schema instance performs book-keeping on its own local blackboard, and 2) the global blackboard is partitioned by object with efficient parallelism in mind. A schema's local blackboard can be located at a node where the schema will generally run. Since there is limited parallelism within a schema, there will be a small amount of potential processor-to-processor communication (or memory contention in a shared memory system). Although in theory any schema can access any section of the global blackboard, in practice most schemas access only a very few sections. Since the set of sections most likely to be accessed is known before a schema is run, we should be able to determine a good distribution of the blackboard information and schemas across several processors.

4. Experimental Results

This section demonstrates the Schema System's current capabilities. Figures 1, 2, and 3 show three images of roads in the Amherst, Mass. area, and their final interpretations. For brevity, we concentrate on roads in this presentation. The system has also been exercised on three house scenes, as well as on three additional road scenes not covered here. The interested reader can find all nine interpretations, with commentary, in [6]. Interpretations are presented here as figures in which *believed* hypotheses (i.e. those with a confidence level of *belief* or *strong-belief*) are shaded according to object; uninterpreted areas of the image are left white. Table 1 shows the shade-to-object code used in presenting the interpretations.

Figure 1 shows a typical road scene containing five discernible objects – road, roadline, road-shoulder, tree trunk and foliage. The scene is deceptively simple since it will foil attempts at interpretation which assume that long, straight lines can be found bounding the road, or that the structural line of a tree trunk can be reliably extracted from the mass of its foliage, or that roadlines will appear as oriented ribbons in the region segmentation. Even for this apparently straightforward example, different sources of information must be combined in object-dependent ways, integrating top-down and bottom-up processing.

Large road regions can be initially distinguished using just color and texture attributes. The road schema verifies their identification by invoking related object schemas such as roadline and road-shoulder; if these produce believed hypotheses that are spatially and geometrically consistent with the road hypothesis, then the road hypothesis gains *belief* status. Furthermore, if a roadline hypothesis extends beyond the currently hypothesized road area, the road schema attempts to extend its hypothesis to match it. Newly hypothesized road extensions which exhibit glaring errors in texture or color are rejected. Accepted road extensions provoke the search for further roadline extensions, and so on, as the road and roadline schemas mutually 'walk' their way down the road. In Figure 1, only the large foreground road region was identified on the basis of color and texture; more remote sections of the road were found in cooperation with the roadline schema.

Figure 2 shows a similar road scene. Note that the warning sign, including its post, has been correctly identified. Finding objects such as sign posts is a good example of the benefits of expectation-driven processing. Searching an image for all mutually overlapping parallel pairs of lines would be prohibitively expensive, even if the lines were filtered first with repect to orientation (in this case vertical). The optic yellow warning sign is fairly easy to pick out, however, and provides strong constraints on the location of its pole.

Figure 3 is an image of a road making an S-curve, and is a good example of the complexity involved in finding even simple objects like painted roadlines. The region segmentation quickly drops the roadline as it recedes into the distance, while the curve of the road breaks up its linear structure. The roadline schema therefore requires both region and line data for its task. The search for roadline is initiated by the road schema, attempting to verify its own hypothesis by finding a roadline subpart. The roadline schema begins a bottom-up search for footholds: reasonable roadline seed hypotheses to pursue. Confining its search to areas of the image near the road hypothesis, the roadline schema looks for narrow (relative to the camera model) regions that match the expected color and texture of roadlines (as determined by the IHS knowledge source which derives these expectations automatically from a set of hand-labeled training images). Only very strong matches are accepted.

Since there is a possibility that the segmentation algorithm has missed the roadline(s) completely, the roadline schema also collects pairs of long, high contrast, mutually overlapping parallel lines. For each pair, a new region is constructed from the pixels lying between the two lines and tested against the color and texture of roadline. This fusion of line and region information allows the system to recover from region segmentation errors.

Once roadline seed hypotheses have been established, a top down examination of the line data is used to extend them. Lines bounding each side of a hypothesized roadline region are taken as starting points for line chains. Based on an ***endpoint-near*** relation, these lines are linked to possible extensions, and the resulting two-line chains are extended in the same manner, forming sequences of lines which are candidates for roadline boundaries. Although line-linking strategies such as endpoint-near occasionally make false connections, as long as the lines in the chains remain mutually parallel there is reason to believe that they are following roadline boundaries. Whenever the roadline schema finds such a pair of piece-wise parallel chains, it constructs the

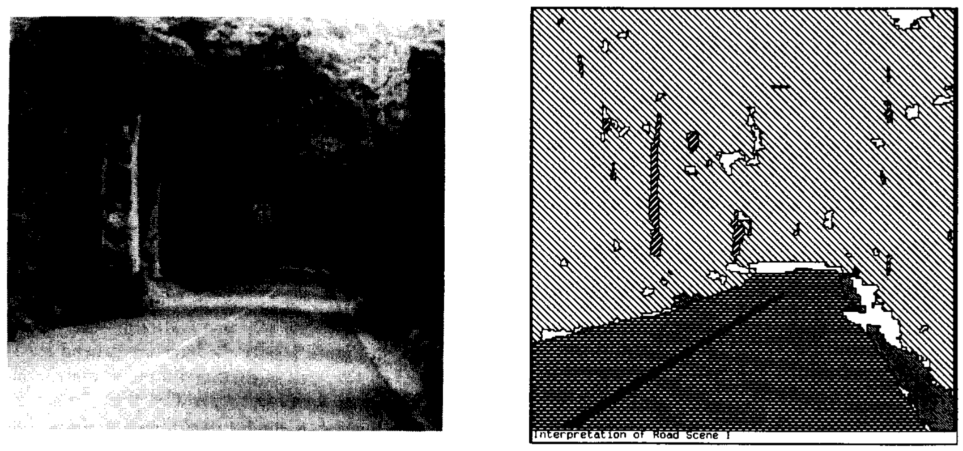

Figure 1: Road Scene 1 and its Final Interpretation

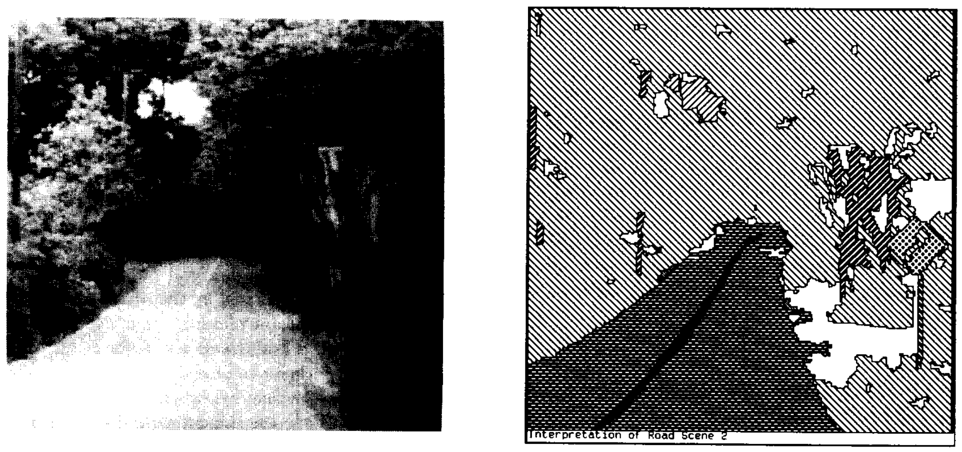

Figure 2: Road Scene 2 and its Final Interpretation

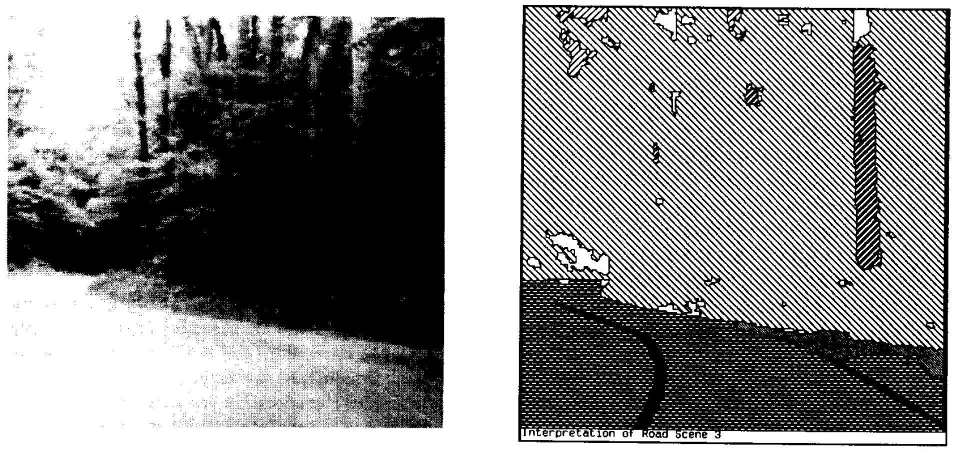

Figure 3: Road Scene 3 and its Final Interpretation

region bounded by the chains and posts it as a *believed* roadline hypothesis. The interpretation in Figure 3 includes two *believed* roadline hypotheses extended in this manner.

5. Conclusion

The UMass Schema System is an ongoing experiment in knowledge-directed vision. Our goal is to achieve the efficiency of a special-purpose vision system while maintaining the breadth of a general-purpose system. To this end, the system has been designed to provide a flexible environment for encoding both object and control knowledge. The general-purpose nature of the system is embodied in the knowledge sources and blackboard kernel routines and representations. Special purpose capabilities are provided by the schemas themselves.

From its inception, the Schema System has been designed to run on a parallel processor. Care has been taken to distribute the vision task and avoid communication bottlenecks. We currently simulate parallelism on a TI Explorer^{TM}lisp machine. While it is imperative that the Schema System be exercised in a truly parallel environment, our target machine, the IUA [23], is still a couple of years from completion. Therefore we have begun to port the system to a Sequent Balance 21000^{TM}multiprocessor. Although this machine is not adequate for real-time vision, it should allow us to test our schema control framework in a parallel, shared-memory environment.

A major question for any knowledge-based vision system is the degree to which its knowledge base can be expanded to accommodate new objects and domains. The Schema System must eventually become sufficiently good at object indexing that only a few, relevant schemas are ever invoked for any given image. In addition, the effects of knowledge base size on communication patterns and control must be determined. Currently, our knowledge bases are inadequate for such an investigation. Serious research into the construction and maintenance of large knowledge bases may begin when a system is able to recognize a large number (50-100) of objects. Toward this end, we are integrating our house and road scene knowledge bases, which will give us a database of about twenty objects. Work on aerial images is being considered to provide insights from a very different domain.

REFERENCES

[1] Michael A. Arbib, "Segmentation, Schemas, and Cooperative Computation", in *Studies in Mathematical Biology, Part 1*, S. Levin (ed.). MAA Studies in Mathematics, Vol. 15, 1978. pp. 118-155?.

[2] Dana H. Ballard, "Model-directed Detection of Ribs in Chest Radiographs," *Proceedings of the Fourth IJCPR*, Kyoto, Japan, 1978.

[3] David Chapman, "Planning for Conjunctive Goals", Artificial Intelligence Vol. 32, (1987), pp. 333-377.

[4] Daniel Corkill, Kevin Gallagher and Kelly Murray. "GBB: A Generic Blackboard Development System" *Proc. of AAAI-86*, Philadelphia, PA, Vol. 2, pp. 1008-1014.

[5] Bruce A. Draper, Robert T. Collins, John Brolio, Joey Griffith, Allen R. Hanson and Edward M. Riseman. "Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes." *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA., Feb. 1987. (Morgan Kaufman Publishers, Inc., New York) pp. 178-193.

[6] Bruce A. Draper, Robert T. Collins John Brolio, Allen R. Hanson, Edward M. Riseman. *The Schema System.* UMass COINS TR, (December 1987).

[7] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser and D. Raj Reddy. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys* 12(2) (June, 1980), pp. 213-253.

[8] Jay Glicksman. *A Cooperative Scheme For Image Understanding Using Multiple Sources of Information.* Technical Report TN 82-13 (Ph.D. thesis), Dept. of Computer Science, Univ. of British Columbia. Nov., 1982.

[9] Allen R. Hanson and Edward M. Riseman. "VISIONS: A Computer System for Interpreting Scenes" in *Computer Vision Systems*, Hanson and Riseman (eds.), Academic Press, New York. 1978.

[10] Allen R. Hanson and Edward M. Riseman. *The VISIONS Image Understanding System - 1986.* COINS Technical Report 86-62, Dept. of Computer & Information Science, Univ. of Massachusetts. Dec., 1986, to appear in *Advances in Computer Vision,* ed. C. Brown, Erlbaum, 1987.

[11] Barbara Hayes-Roth "A Blackboard Architecture for Control", *Artificial Intelligence* 26(3) (July 1985), pp. 250-321.

[12] Shang-Shouq Vincent Hwang, *Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding.* Ph.D. thesis, Dept. of Computer Science, Univ. of Maryland. 1984.

[13] Philip M. Johnson, Daniel D. Corkhill, and Kevin Q. Gallagher. "Integrating BB1-Style Control into the Generic Blackboard System," presented at the *AAAI-87 Workshop on Blackboard Systems*, Seattle, Washington, July 1987.

[14] Richard E. Korf, "Macro-operators: A Weak Method for Learning", *Artificial Intelligence*, Vol. 26 (1985), pp. 35-77

[15] David M. McKeown, Jr., Wilson A. Harvey, Jr., and John McDermott, "Rule-Based Interpretation of Aerial Imagery", *IEEE trans. on Pattern Analysis and Machine Intelligence*, 7(5) (Sept. 1985). pp. 570-585.

[16] Marvin Minsky "The Society Theory of Thinking" in *Artificial Intelligence: An MIT Perspective*, P.H. Winston and R.H. Brown (eds.). MIT Press, Cambridge. 1979. Vol 1., pp. 423-450

[17] Makoto Nagao and Takashi Matsuyama. *A Structural Analysis of Complex Aerial Photographs.* Plenum Press, New York. 1980.

[18] Yu-ichi Ohta. *A Region-Oriented Image-Analysis System by Computer.* Ph.D. thesis, Dept. of Information Science, Kyoto University. March, 1980.

[19] Steven A. Shafer, Anthony Stentz and Charles E. Thorpe. "An Architecture for Sensor Fusion in a Mobile Robot," *Proceedings IEEE International Conf. on Robotics and Automation,* San Francisco, California, pp. 2002-2011, 1986.

[20] John K. Tsotsos, "Knowledge Organization and Its Role in Representation and Interpretation for Time-Varying Data: The ALVEN System," *Computational Intelligence,* Vol. 1, 1985, pp. 16-32.

[21] Lewis W. Tucker, *Computer Vision Using Quadtree Refinement,* Ph.D. dissertation, Polytechnic Institute of New York, May 1984.

[22] Charles C. Weems, Steven P. Levitan, Allen R. Hanson, Edward M. Riseman, J. Gregory Nash, and David B. Shu, *The Image Understanding Architecture,* COINS Technical Report 87-76, University of Massachusetts at Amherst, 1987.

[23] Terry E. Weymouth, *Using Object Descriptions in a Schema Network For Machine Vision,* Technical Report 86-24, (Ph.D. thesis) Dept. of Computer and Information Science, Univ. of Massachusetts, May 1986.

SESSION 1.3 — STEREO

Chair

M. Trivedi
University of Tennessee