TOOLS AND EXPERIMENTS IN THE KNOWLEDGE-DIRECTED INTERPRETATION OF ROAD SCENES

Bruce A. Draper, Robert T. Collins
John Brolio, Joey Griffith
Allen Hanson, and Edward Riseman

COINS Technical Report 87-05

January 1987

Abstract

The *Schema System* is a high-level image understanding system for the interpretation of complex natural scenes. It is our position that the constraints available from general knowledge about the world must be used to organize raw image descriptions into abstract interpretations. The output of low-level image operations does not meet the requirements of natural-scene interpretation. Intermediate-level grouping procedures represent a partial solution to this mismatch, but are too expensive to apply indiscriminately. High-level processes use object-specific knowledge to guide interpretation by focusing attention on promising areas of the image. This paper describes the Schema System - a flexible, high-level system supporting the interaction of object-specific interpretation processes - and presents the results it has produced on road scenes as a justification of the knowledge engineering approach to image understanding.

This research was supported by DARPA under contracts N00014-82-K-0464, DACA76-85-C-0008, NSF DCR-8500332, and by AFOSR under contract F49620-86-C-0041.

1 Introduction

The University of Massachusetts *Schema System* is a high-level, knowledge-directed image understanding system for the interpretation of complex natural scenes. It is our position that the constraints available from general knowledge about objects and the structure of typical scenes can be used to develop abstract interpretations from raw image descriptions. Low-level image operations such as straight line extraction and region segmentation are meant to be general enough to serve a broad range of intermediate and high-level requirements. Consequently, their output is relatively unorganized and does not necessarily reflect the requirements of natural-scene interpretation. Intermediate-level grouping procedures organize low-level data in ways more suitable for interpretation. This is a partial solution, but many of these processes are too expensive to apply indiscriminately, and are not relevant in all situations. High-level processes use real-world knowledge to hypothesize likely objects and manage levels of belief about these hypotheses. They guide the interpretation by focusing attention on promising areas of the image and allocating resources to context-specific strategies as they become appropriate. As the interpretation proceeds, the high-level components provide feedback to low-level processes to tailor their output to the type of image and to the needs of the interpretation (see Figure 1).

With this scenario in mind, we have developed the Schema System as a flexible, high-level system to support and control the interaction of object-specific interpretation processes. This system is currently operating in conjunction with the UMass Low Level Vision System (LLVS) to identify and extract the significant objects in natural scenes. This paper describes the theory and practice of the Schema System; its architecture, its database organization and interpretation techniques, and the results it has produced in the domain of New England road scenes. We present the experimental results as a justification of the approach, and to demonstrate that the required knowledge engineering task is feasible.

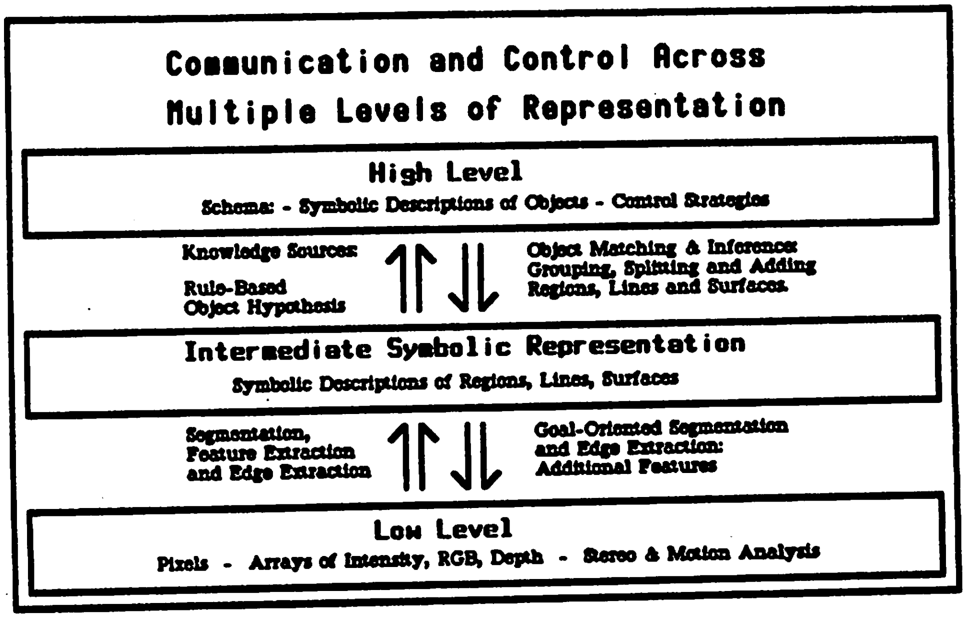

Communication and Control Across
Multiple Levels of Representation
High Level
Schema: - Symbolic Descriptions of Objects - Control Strategies
Knowledge Sources:
Rule-Based Object Hypothesis
Object Matching & Inference: Grouping, Splitting and Adding Regions, Lines and Surfaces.
Intermediate Symbolic Representation
Symbolic Descriptions of Regions, Lines, Surfaces
Segmentation, Feature Extraction and Edge Extraction
Goal-Oriented Segmentation and Edge Extraction: Additional Features
Low Level
Pixels - Arrays of Intensity, RGB, Depth - Stereo & Motion Analysis

Figure 1: Multiple levels of communication.

1.1 Overview

The task of high-level vision is to identify meaningful objects and relations in natural scenes, and ultimately to generate a three-dimensional interpretation. Meaningful objects include those which a person might identify from the image, as well as objects which must be identified for a particular task such as navigation. Meaningful relations are those spatial relationships which serve to locate objects relative to each other, and to locate the scene relative to the observer. A high level vision system should identify the semantic objects in the scene and gather information about them.

One key to the construction of an image understanding system is the use of general knowledge about the world to guide the interpretation of image events. Low-level processing on a typical natural scene produces an extremely large number of *tokens*: thousands of lines, hundreds of regions, and an arbitrary number of other detectable events. Not all of these tokens are significant. Intermediate-level grouping processes partially alleviate this problem by combining the low-level data into a smaller number of relatively salient events. However, these processes are typically too expensive to apply everywhere. Moreover, they are subject to the same problems that afflict the low-level processes – the relevance and proper organization of the events they detect are dependent on the context in which they occur. High-level control is needed to focus such processes on a limited number of events under the appropriate circumstances.

In addition to focusing the attention of lower level processes, a high-level vision system must evaluate the extracted events in terms of its interpretive context. For example, merging adjacent regions depends on more than their measurable attributes in feature space. The decision should also be based on the image characteristics of the object involved – high contrast lines are commonly found on the texture boundaries of foliage regions, and thus should not inhibit region merging if the entire tree crown is desired, whereas the presence of high contrast lines along the edge of a tree trunk region generally signals the boundary of the trunk. Object-based hypotheses also provide access into a library of object-specific interpretation strategies.

1.2 The System

The Schema System applies real-world knowledge to the interpretation of natural scenes. The knowledge base is composed of *schemas* and *knowledge sources.* A *schema* is a template for recognizing some class of objects ("object" is used in a general sense that includes road scene and outdoor scene). A schema can be thought of as a frame, packaged with its own interpreter. When the current context predicts that a given object instance might be present, a new schema instance is *instantiated.* A schema instantiation is a copy of the schema template which runs as an independent agent with its own process and local state. The instance processes image data concurrently with previously activated schema instances, and can communicate with them through a central blackboard. A schema instance has two primary tasks, *control* and *synthesis.* Control involves the sequence of actions that a schema instance will take, such as determining which knowledge sources to run, and when and where to run them. Control also involves dynamic resource allocation, which we have not yet experimented with. Synthesis is the process of integrating knowledge source results into a single hypothesis, and linking individual hypotheses into a consistent interpretation network.

Knowledge sources are image interpretation utilities available for invocation by schema instances. They are generic tools for performing common actions such as graph matching and feature comparison. They have a simple parameterized control structure, and they maintain no internal state between invocations. Two knowledge sources are discussed later in this paper: Rulesys, a feature-based initial hypothesis generator, and OVARC, a system for *O*bject *V*erification by the *A*pplication of *R*elational *C*onstraints.

1.3 The Application of Knowledge

During an interpretation, schema instances will employ many different types of knowledge. The types of knowledge currently used include: 1) two-dimensional and three-dimensional models (e.g., telephone pole or road sign), 2) feature space characteristics (initial hypotheses), 3) part-subpart and ISA relations (road sign/pole, road/roadline), 4) co-occurrence heuristics (gravel next to road), and 5) relative size, position, and location (sequences of telephone poles lining the road). The level of description that this knowledge is concerned with is much higher than the level of abstraction at which low-level processes operate. Central to the system, therefore, is a database called the Intermediate Symbolic Representation (ISR). The ISR represents image data in terms of symbolic *tokens*. The simplest tokens are output by the low-level line extraction and segmentation algorithms (other forms of low-level data, representing depth, motion, etc., can also be stored). Others are "hallucinated" by high-level processes. The most sophisticated tokens are typically defined in terms of other tokens, by iterative processes that build more and more abstract tokens from the database, as in the rectilinear line grouping system of Reynolds and Beveridge [Reyn87].

The interpretation process proceeds opportunistically. The system has certain initial expectations, such as being outdoors and on a road, so the system is started with outdoor scene and road scene schema instances already active. These schemas in turn predict the presence of objects and start schemas to recognize them. Schemas that fail to generate initial hypotheses terminate; the rest attempt to verify their hypotheses, possibly predicting new objects and starting new schema instances along the way. This leads to a combined bottom-up and top-down interpretive approach, in which characteristics of objects that are cheap to compute are used to generate initial hypotheses, and the attempt to verify or negate these hypotheses then drives further processing. Schema instances whose objects are complementary communicate this to each other as a source of support. Those whose interpretations contradict each other compete. The final interpretation emerges as a network of consistent and believed object hypotheses.

The remainder of this paper presents the major components of the system in more detail, and describes a set of experiments using this system to interpret New England road scenes. The Schema Shell, the ISR, and the Rulesys and Object Verification (OVARC) knowledge sources are each presented in greater detail. Then the results are presented, including details of some of the schemas and knowledge sources used, and traces of intermediate results showing how the final interpretation was assembled.

2 Tools

2.1 The Schema Shell

The *Schema Shell* is a tool that supports the construction of large sets of schemas [Drap86]. The road scene interpretation schemas presented in this paper form one such set; a set of schemas for house scenes was previously developed. The shell provides a simulated distributed environment (until parallel hardware arrives) in which any number of schema instances may run concurrently.

Communication between instances is implemented through a central *blackboard*. While each schema instance executes in its own memory space, all instances may write to and read from the blackboard. This provides a uniform, asynchronous communication mechanism between schema instances.

The notion of schemas supported by the Schema Shell is the result of a long line of research at the University of Massachusetts [Arbi78], [Arbi81],[Hans78],[Weym86]. The earliest work was biologically motivated and theoretical in nature [Arbi78], [Arbi81]. The first implementations of schemas, and the first use of schemas in vision, were similar to a frame based system [Hans78]. More modern implementations have focused on the aspects of concurrency and the active nature of the agents [Weym86]. Schemas as defined by the Schema Shell are both a simplification and a generalization of the concept used by Weymouth. Major differences include the adoption of a single, general purpose communication mechanism, the separation of local memory from communication, the formalization of the notion of strategy, and a simplified method of schema creation.

A schema requires local memory and a process to operate on it. The former is specified by the user as a list of schema variables; these may either have initial, object-specific values (such as a two-dimensional model for the OVARC knowledge source) or may be left to be filled in with image-specific data. The latter is provided by the user in the form of an *interpretation strategy*. This is a Lisp procedure which invokes knowledge sources in response to the current context, and which stores the resulting data in the schema variables. When it is possible to perform aspects of the recognition task in parallel, the strategy may spawn other strategies which operate concurrently on the same local memory. This provides another level of parallelism below the schema level. In the experiments described below, each schema was in fact implemented as up to seven independent strategies. When a schema instance predicts the presence of a related object in the image, it invokes a new schema instance which is allotted a distinct section of memory. The new instance pursues its hypothesis independently of its parent. In this way, the system will have many concurrently executing schemas, and within each schema, several concurrent strategies.

Schema instances communicate through a central blackboard. At any point during its processing a schema strategy may write an arbitrary message to the blackboard. Any other schema is then free to read, erase or modify that message. This provides a single, uniform communication mechanism which can be easily implemented on a variety of distributed architectures. To post a message, a schema instance needs only a section name to write to. The mechanisms for reading, erasing and modifying messages are of necessity more complex. To read a message, a schema instance must specify the section to be read from, and some method for selecting among the messages present on that section. The selection criterion is provided in the form of a predicate; all messages for which the predicate evaluates *true* are returned by the read function. Time of posting may also be used as a selection criterion. If no messages are found, there are two possible actions. The read function can return null, or it can suspend the calling strategy until a suitable message is posted. Since both behaviors are useful, the Schema Shell provides two basic reading functions, *read-or-nil* and *read-or-wait*. If no messages are found, the former returns null, while the latter suspends the caller. A suspended instance is said to be *sleeping*, and consumes virtually no resources until a suitable message is written and the instance is *awakened*. The shell provides routines for removing messages from the blackboard as they are read (*erase-or-nil* and *erase-or-wait*) and for altering existing messages (*modify-blackboard*).

It should be noted how this form of communication differs from that of both object oriented message passing systems and traditional blackboard based systems. In most message passing systems, there is a strong tie between the sender and receiver of a message. The sender must know the identity (i.e. unique address) of the receiving instance in order to send the message. In addition, some systems suspend the caller until the receiver returns a value. In the Schema Shell this bond is severed. The sender is not required to know the identity of the receiving process. It writes the message to the blackboard, and relies on the receiving instance (or instances) to read it. Messages are also received differently than in traditional systems. In most message passing systems, the receivers act as slaves (for a counterexample, see [Hewi77]); they perforce take the action dictated by the message. Schemas, on the other hand, must choose to read the message. Thus, communication is a cooperative venture rather than a master/slave activity.

The differences between the Schema Shell's message passing and the traditional variety stem from a different style of computation. Object oriented systems are generally hierarchical, while the Schema Shell and other blackboard based systems are heterarchical. Thus, communication in object oriented systems takes place between closely related objects, in which the sender is commanding the execution of subtasks. Message passing in the Schema Shell, on the other hand, is between arbitrary, independant schemas. As such, it is usually a way of sharing (often uncertain) information rather than distributing subtasks. In this way, the goals of inter-schema communication closely resemble those which motivated the development of traditional blackboard systems ([Erma80]).

At the same time, the Schema Shell differs from traditional blackboard systems in that the blackboard performs only synchronization and data buffering, not control. In Hearsay-II [Erma80] data events trigger knowledge source execution through a blackboard-based scheduler. In the Schema Shell knowledge sources are selected by schema instances according to the interpretive context. Invocation of knowledge sources and schema instances is completely indepedent of any blackboard mechanism. In both the Hearsay-II and Schema Shell paradigms, the knowledge sources are uninterruptable, and maintain no state between invocations. In Hearsay-style systems, all data must be kept on the blackboard, presenting a possible bottleneck. In the Schema Shell, local object hypothesis information is maintained within the Schema instance itself. Only data that needs to be shared between instances is posted to the blackboard. This has two advantages: first, the bandwidth of communication across the blackboard is kept small; second, instance specific data is kept private. In this last respect, the shell is similar to some other modern blackboard systems, e.g. [Shaf86].

The Schema Shell additionally provides I/O routines and debugging aids. These include tools for identifying the status of schema strategies, inspecting schema instances' local memory, and displaying ISR tokens to either black-and-white or color monitors. Using a mouse-driven display during debugging helps the programmer to visually locate and identify a desired token from among hundreds.

2.2 The Intermediate Symbolic Representation

Input to the Schema System consists of abstract image descriptions which are produced from low-level processes of line and region extraction, and from intermediate-level processes of grouping and selection. These image descriptions are stored in the Intermediate Symbolic Representation. The ISR is a database which has been custom-built for the efficient storage, manipulation, and retrieval of abstract image data. The fundamental unit of representation is the *token*. Each token has a unique name and a list of feature slots. The ISR can be used to store anything that can be characterized by a list of features and values; some of the image events currently stored include region segments, extracted edge lines, fields of homogeneous texture, rectilinear line groups, and region-line relations. The benefits which result from imposing a uniform representation and user interface on all intermediate level tokens are enormous. It is now natural to think in terms of multistage and hierarchical grouping processes which take in tokens at one level of abstraction and produce tokens at the next higher level [Reyn87], [Weis86]. The freedom from having to represent image events by sets of pixels makes it easier to store relational tokens, that is, tokens which represent the relationships between other tokens [Belk85]. The ISR also facilitates the sharing of results between researchers, and between systems – it is implemented in both the LLVS and the old VISIONS system, and results produced by one can be read in to the other thanks to the ISR's standardized data format.

An organized environment is maintained in the ISR by partitioning the total set of tokens. At the highest level, tokens are partitioned according to the *image* they were extracted from. There is also an intermediate level of partitioning called the *tokenset*. All tokens in a tokenset must have the same features defined for them, so in a sense they are all of the same "type." For instance, a standard line tokenset contains features which have meaning when applied to lines, like length, orientation, and contrast. Tokens in a region tokenset have different, region-specific features, like average intensity, compactness, and Euler number. By convention, tokensets contain all the tokens produced by the same abstraction process running on a single image, but tokensets can also be used to impose arbitrary groupings of tokens of the same type, such as dividing them up by spatial location.

Each feature associated with the tokens in a tokenset has a name, a value slot, a data type, and a computation function. Supported data types include numeric fixed and floating point, as well as a pointer data type for representing strings or lists of other tokens. Since many tokens have a physical realization in an image, a special bitplane data type is provided for representing the subset of image pixels which are associated with a token. Operators exist for taking the intersection, union, and difference of token bitplanes; bitplanes can also be created from scratch by specifying a list of bounding vertices. This directly supports strategies for fusing information across multiple representations. For example, relations between lines and regions can be derived by creating line bitplanes and intersecting them with region bitplanes and analyzing their overlap [Belk85].

Two of the more interesting aspects of the ISR involve the way features are accessed; these methods include associative access and on-demand computation. Associative access complements the normal data access mechanism (which can be loosely translated as "give me the length of line 45 in image 1") by allowing the ability to access tokens by attribute ("give me the lines in image 1 whose length is between 5 and 10"). The present associative capabilities include coarse spatial location ("give me the lines in image 1 whose bounding rectangle overlaps X", where X is a specification of some rectangular window in the image), conjunctive filtering on ranges of numeric feature values ("give me the long, high contrast lines"), and limited disjunctive filtering ("give me the long, high contrast lines with an orientation of close to 90 or close to 180 degrees" – that is, vertical and horizontal, long, high contrast lines).

The other interesting access mechanism in the ISR is on-demand feature computation. When a feature with an undefined value is accessed, the computation function associated with that feature is invoked. The function may choose to store the value, along with any intermediate or related values that have been computed, in which case the value is retrieved directly the next time the feature is accessed. If the value does not get stored, that function will be invoked again when the feature is accessed again. Whether or not a value was directly retrieved or had to be computed is transparent to the user (other than in terms of visible speed of the process). This facility for on-demand feature calculation has three important benefits. One, only the features that are actually going to be used need to be calculated; two, it is possible for a schema instance to define new features or redefine old ones dynamically during the course of an interpretation; and three, new tokens can be introduced into the database at runtime, and they will be essentially indistinguishable from pre-existing tokens.

This last point bears repeating. The ISR allows schema instances to create tokens during an interpretation, create their bitplanes either from scratch or as some combination of token bitplanes, and access their features, at which time the new feature values will be calculated automatically. It is thus possible for the schema system to dynamically "correct" misleading segmentations based on combinations of top-down knowledge, and to "hallucinate" regions based on structures found in the line data.

2.3 The Rulesys Knowledge Source

A high level vision system requires at least one component knowledge source for comparing measurable attributes of image events to expected attributes of known objects in the world. At present, the Griffith Rule Generation System, or Rulesys knowledge source, fills that role for the Schema System. The Rulesys knowledge source constructs rules which map feature values into *object like-*

lihood scores. The resulting scores suggest initial hypotheses which the Schema System considers when evaluating promising paths of inquiry.

The Rulesys applies an automatic rule construction method similar to [Lehr87] and [Belk85]. The input to the system is a list of significant objects, a set of training images in which those objects have been hand-labelled, and a list of the measurable attributes over which the rules will be defined. The resulting rules distinguish one object from another in the knowledge base. The Rulesys knowledge source operates in an environment of unreliable feature measurements and imperfect *a priori* information about object attributes. To ensure robustness, many partially redundant features combined to provide more reliable results. A simple rule maps single feature values into object likelihood scores. Complex rules group simple rules into general classes such as color and texture. Scores are combined to produce a single score which relates an object class to a feature category. Currently implemented feature categories are color, texture, shape, size, and location. For the New England road scene experiments presented in this paper, only color and texture rules were used.

2.4 The OVARC Knowledge Source

Once the Schema system has established a set of initial hypotheses, it is necessary to invoke verification strategies to confirm or reject the labelling proposed by each hypothesis. These strategies in turn rely on knowledge sources (KSs) which are general-purpose modules whose output will affect the system's confidence in a hypothesis. One of the knowledge sources we have been experimenting with performs object verification by the application of relational constraints (OVARC). The purpose of OVARC is to take a structure or template called an object description (OD) and to find a set of tokens which match the template, attribute for attribute, relation for relation. The attributes (*1-place predicates*) and relations together constitute the constraints on the tokens which define an instance of the OD in the image.

An OVARC object description is a graph. Each vertex of the OD represents a token. Attribute constraints are represented as an arc from the vertex to itself. Relational constraints are represented as a directed arc from one vertex to another in the graph. The name of an arc is either the name of an attribute or relation stored in the database (ISR) or of a function which computes the constraints as each candidate data set is processed. This gives the user a great deal of flexibility in the choice of constraints for the object description. The description may contain any relation or constraint which is represented in the database or which can be computed from the database or from the interpretation context. The object description structure controls the allocation of processing resources by defining the order of application of constraints.

The output of OVARC is either a signal of failure to match or a new aggregate token which represents the object defined by the OD. This new token will have attributes and relations of its own, one of which is the internal structure which qualifies it as a match to the OD.

The input tokens for an OD match may be of different types (regions, lines, groups, other objects) or they may all be of the same type. It is possible to have a description of an outline view of a telephone pole or road sign (lines) or a description which includes a piece of road (region), pieces of road edge (lines) and pieces of a centerline (lines and region).

The final step in the matching process is a subgraph isomorphism search [Ullm76]. Taking a set of data tokens with their attributes and relations as the candidate data graph, the goal is to find one (or more) subgraphs which are isomorphic to the object description graph. All solutions to the subgraph isomorphism problem exhibit exponential behavior. Essentially we have taken a method which is exponential in n, but which is otherwise adequate in expressiveness and generality, and restricted its use to situations in which n is very small. In the worst case the complexity of the subgraph isomorphism algorithm is $\binom{m}{n}^n$ where n is the number of vertices in the object description graph and m is the number of nodes in the candidate data graph. For this algorithm to perform reasonably the number of vertices in the OD should be very small (less than 10, preferably less than

6) and the constraints in the OD should reduce the candidate data graph to a relatively small size as well ($\sim n^2$). The size restriction on the OD graph is not a serious restriction since a complex object description can be decomposed into a hierarchical set of ODs.

Given an OD of the proper size, it is necessary to reduce the size of the candidate graph. Since OVARC is an object verification KS, it will be invoked only when there is a reasonable likelihood of finding the object, so the search space will already have been reduced from the whole image to a few locations in the image. Furthermore, in creating an OD, a Schema designer has a great deal of flexibility in controlling computation. To begin with, there are several preliminary filters which can be invoked to reduce the search space. Before the matcher is run, every effort is made to reduce the size of the candidate graph. For each vertex in the OD, constraints are specified which reduce (1) the number of tokens which can match this vertex (via attribute constraints) and (2) the number of tokens which are candidates for related vertices (via relational constraints). The vertices are processed strictly according to the order specified in the OD, so that knowledge of the relative strength and computational cost of individual constraints can be reflected in the order of their application. Vertices with strong constraints are listed first in the description. Vertices with constraints which require more computational resources are listed later when presumably fewer candidate tokens will have survived to be processed. Due to the progressive filtering of the candidates, constraints will be computed only for those candidate tokens which have some chance of matching vertices in the OD. Proper timing of these calculations gives a researcher the ability to fine-tune the description to get the best matching behavior for the smallest amount of computation. The calculation of some constraints may be so costly that an OD will defer their computation until matches have been found. A post-filtering facility is included for this purposes.

Since the heart of this object verification KS is an algorithm with exponential behavior, after all elimination of candidate vertices is complete and the matching process is about to begin a complexity check is made. If the size of the candidate set exceeds a user-specified limit, OVARC reports a *computational complexity failure*. This failure can be interpreted in one of two ways: (1) either the object description is too weak, i.e., it admits too many potential matches, or (2) this knowledge source is not able to handle this particular set of data. The Schema system should be able to use this information to dynamically adapt its interpretation methods. Failures of type (2) are closely related to "normal" failure, i.e., failure to match the object description. A successful match is very strong support for the presence of a given object in the image, but a failure to match may mean the object is absent or that it is present but occluded or viewed at an unusual angle. The only convincing evidence that an image event does not represent some particular object is a verification that it in fact represents a different object.

We are currently expanding the set of descriptions and the capabilities of the description language. One feature which appears to be useful is the ability to specify "non-unique" vertices in the object description. For example, one of the line-extraction operators that we use tends to fragment lines such as the crossbar edges in a telephone pole (see example OD). A description could specify fragmented collinear lines with a "could-also-be-vertex-i" relation, so that two vertices could each represent a token (i.e., there are two line fragments) or both vertices could be matched by the same token. In general, however, matching partial or incomplete views is part of the larger high-level vision problem. The OVARC knowledge source provides some flexibility in describing objects, but further research in grouping will have to uncover ways of conveying information from good partial matches to higher-level processes.

3 Experimental Results

The aim of this experiment was to demonstrate the feasibility of the knowledge based approach to vision by showing that 1) constructing a knowledge base for a domain is a manageable task, and 2) the use of this knowledge produces significantly improved image interpretations. Section 3.1

discusses the knowledge base developed for interpreting road scenes. It is the product of a small ($2\frac{1}{2}$ man) development team working for approximately one month. The team had available to them knowledge sources from the house scene domain; the assembly of a library of useful knowledge sources from many different domains is an ongoing aspect of this work. Section 3.2 describes how the schemas outlined in 3.1 produced the interpretation results shown below.

There are a total of six road scene images. In all six, the camera was level to gravity. In five of the six, it was positioned as a vehicle in the road. Figure 2 shows a sample scene. The poorly defined road edges, heavy shadowing and extreme depth of field are all typical of the domain. Figure 3 is an image from the set which has an uneven ground plane. This complicates 2D interpretation, and will present a challenge to future 3D reasoning extensions. Finally, Figure 4 shows an image with more man-made structures. Interpretation results for these images are presented in Figures 5a and 5b (results for Figure 2), 6a and 6b (results for Figure 3), and 7a and 7b (results for Figure 4).

There are two interpretation errors due to incomplete schema knowledge. In Figure 7a, the detection of the roof supplies enough information to identify the house wall below, but there is no building schema to accomplish this. In Figure 7b, the bottom of the telephone pole is incorrectly labelled tree trunk. This error will be corrected by an object-completion knowledge source that extends the telephone pole model after OVARC has found it. This is easily done by following the linear structure of the pole.

3.1 The Knowledge Base

There are currently 12 classes of objects in the knowledge base: sky, foliage, tree trunk, road, roadline, roadside gravel, telephone pole, telephone wire, roof, stop sign, yellow (caution) sign, and road scene. A decision was made not to include field or grass, since only one instance of each appears in the image set. The objects are organized in a *calling hierarchy* (Figure 8) which

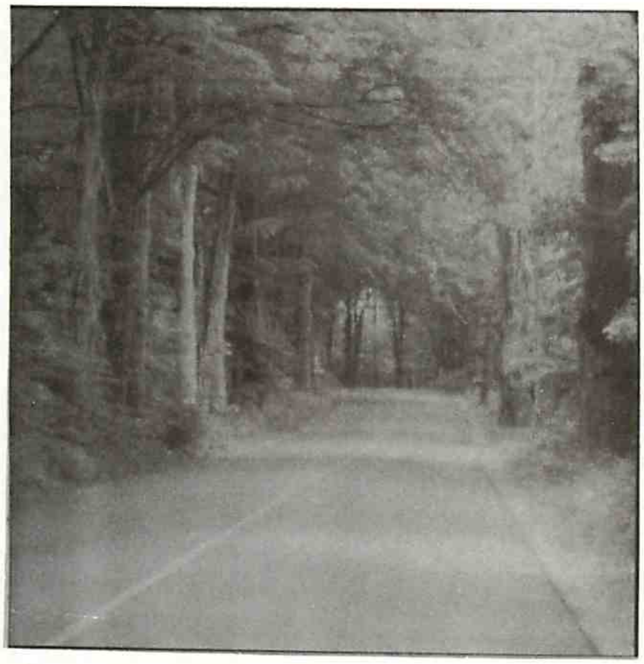


Figure 2. Sample road scene photograph.

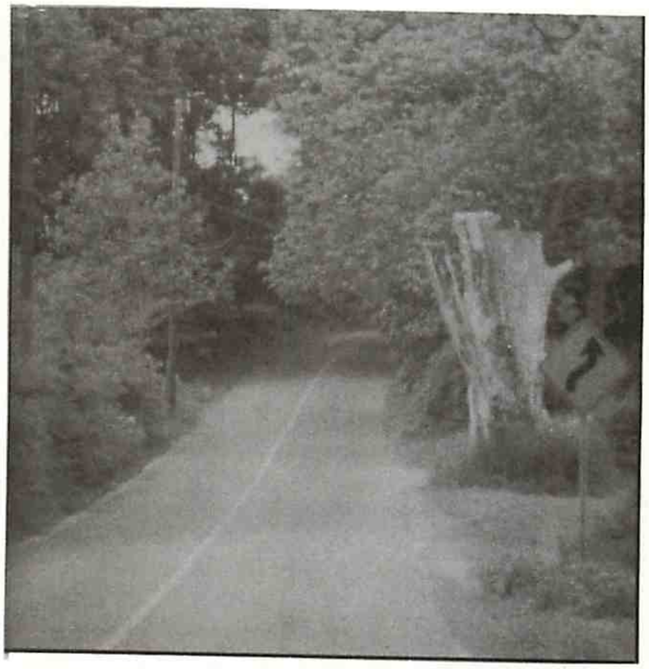


Figure 3. Road scene with an uneven ground plane.

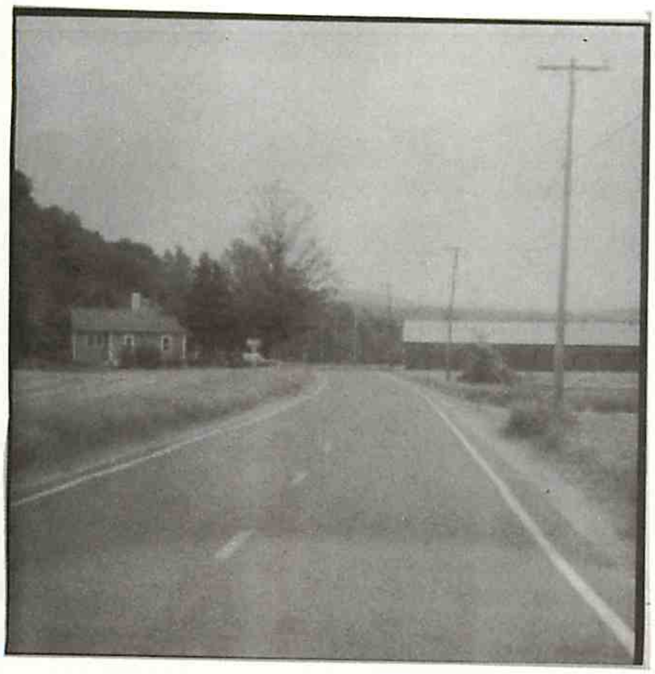


Figure 4. Road scene with other man-made structures.

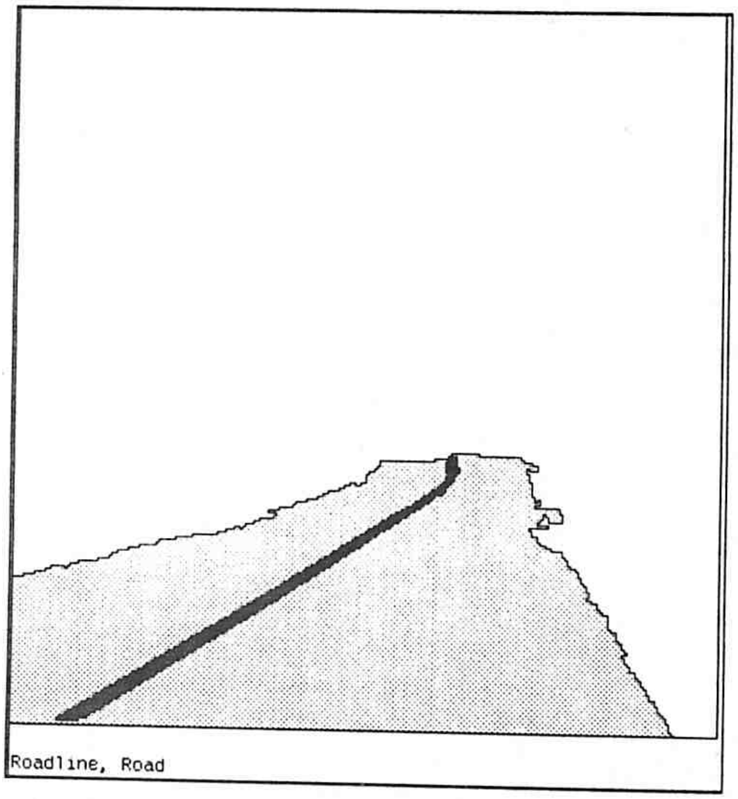
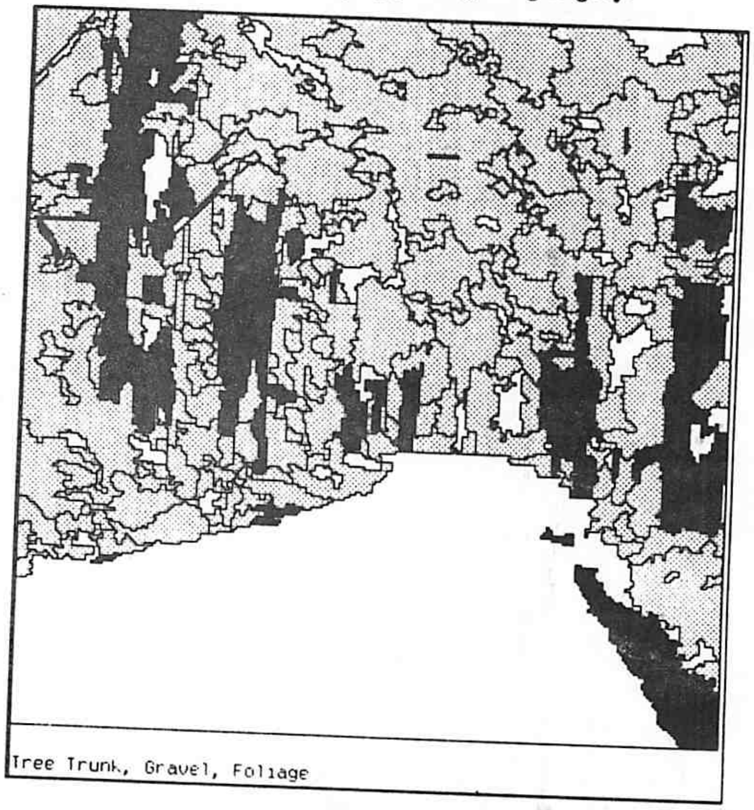


Figure 5a,b. Interpretation results for Figure 2. (a) Road line: black; road: light grey. (b) Tree trunk: black; gravel: dark grey; foliage: light grey.



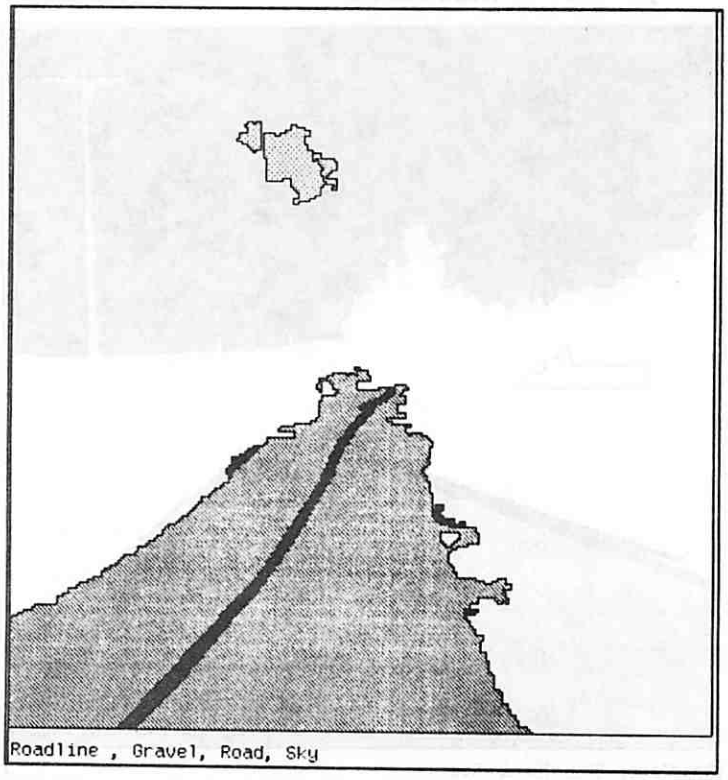
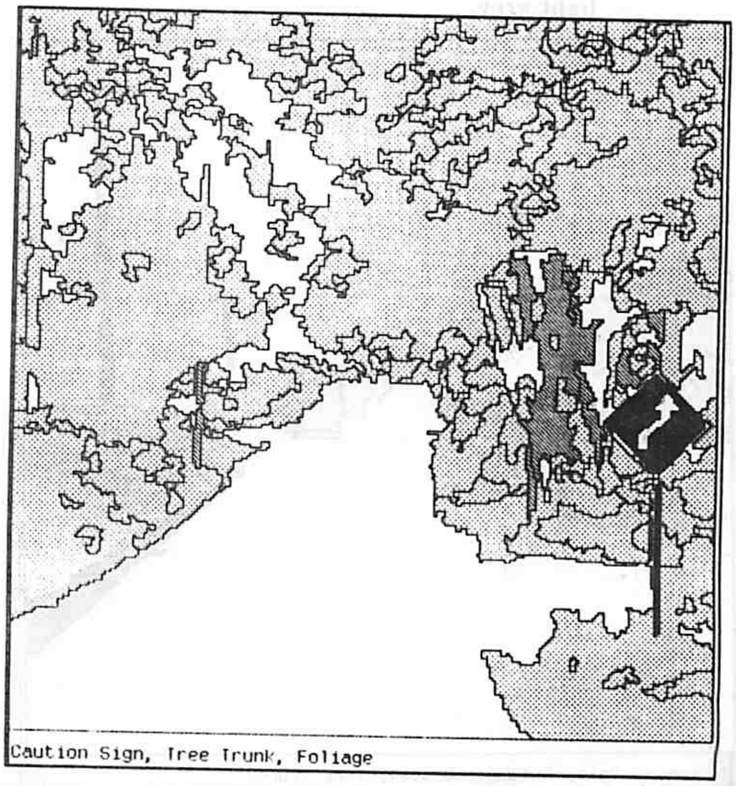


Figure 6a,b. Interpretation results for Figure 3. (a) Road line: black; gravel: dark grey; road: medium grey; sky: light grey. (b) Caution sign: black; tree trunk: medium grey; foliage: light grey.



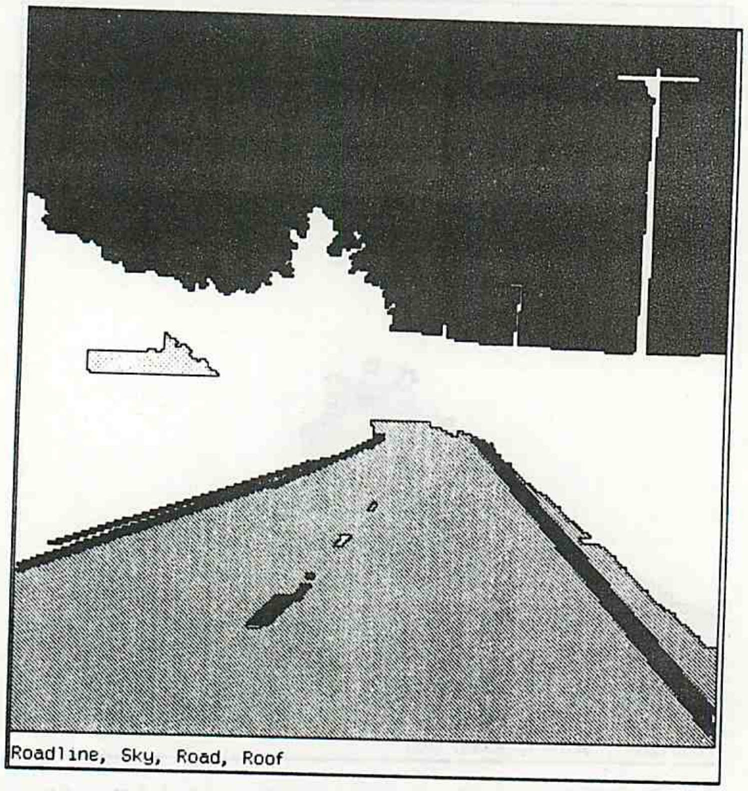
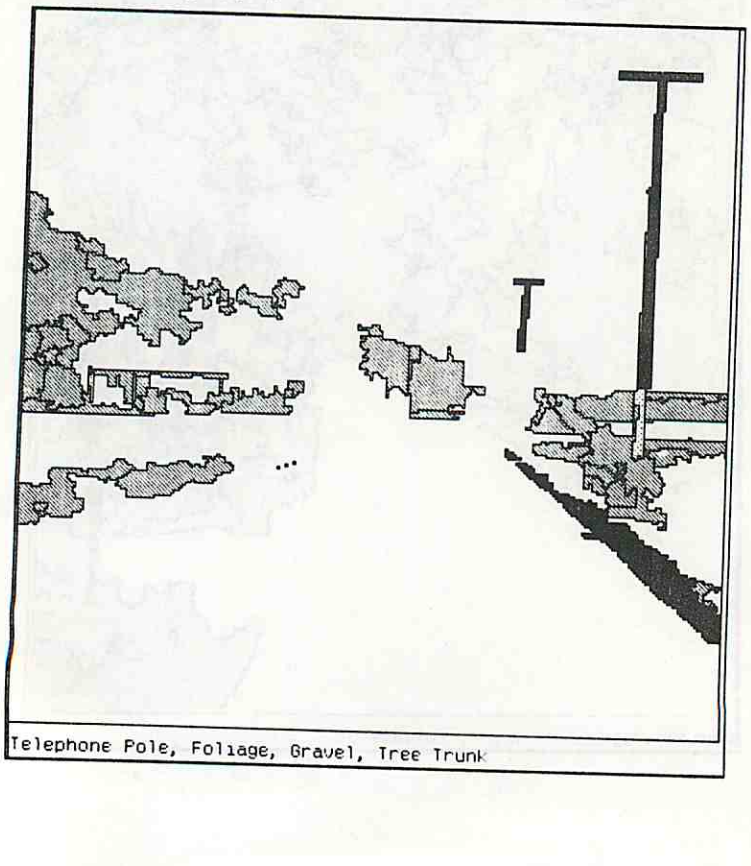


Figure 7a,b. Interpretation results for Figure 4. (a) Road line: black; sky: dark grey; road: medium grey; roof: light grey. (b) telephone pole: black; gravel: (dark grey): foliage: medium grey; tree trunk: light grey.



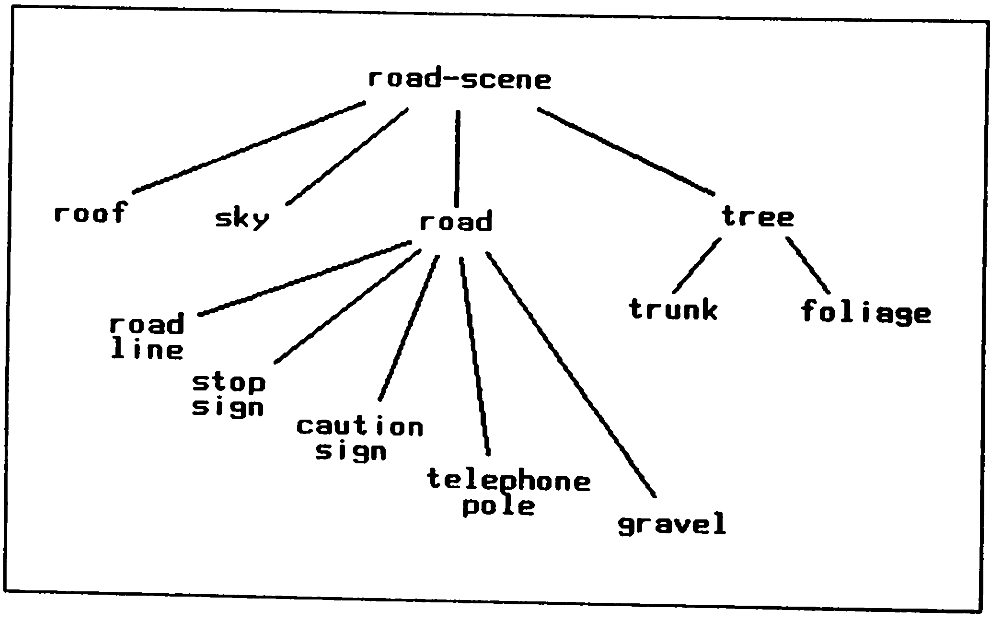


Figure 8: Schema calling hierarchy.

indicates which schemas may invoke what other schemas. This generally follows the traditional part/subpart hierarchy, but it is not compelled to do so. Any information about one object that can be used to predict another should generate a connection in the calling hierarchy between the two objects. In this experiment, the calling structure was strictly hierarchical; this avoided problems with generating redundant schema instances. Our impression is that future knowledge bases will require a tangled calling hierarchy.

How to combine evidence from different knowledge sources and propagate evidence between objects is a difficult, unanswered problem. In this experiment every schema maintained both an internal and an external hypothesis. The internal hypothesis is kept in the schema's local memory, and is used for recording object specific symbolic endorsements. The external hypothesis resides on the blackboard, and has a confidence value on a simple, five-valued scale, **no-evidence, *slim-evidence, *partially-supported, *belief* or **strong-belief*. The endorsements of the internal hypothesis are symbolic results returned by knowledge sources. The road schema, for example, recognizes the

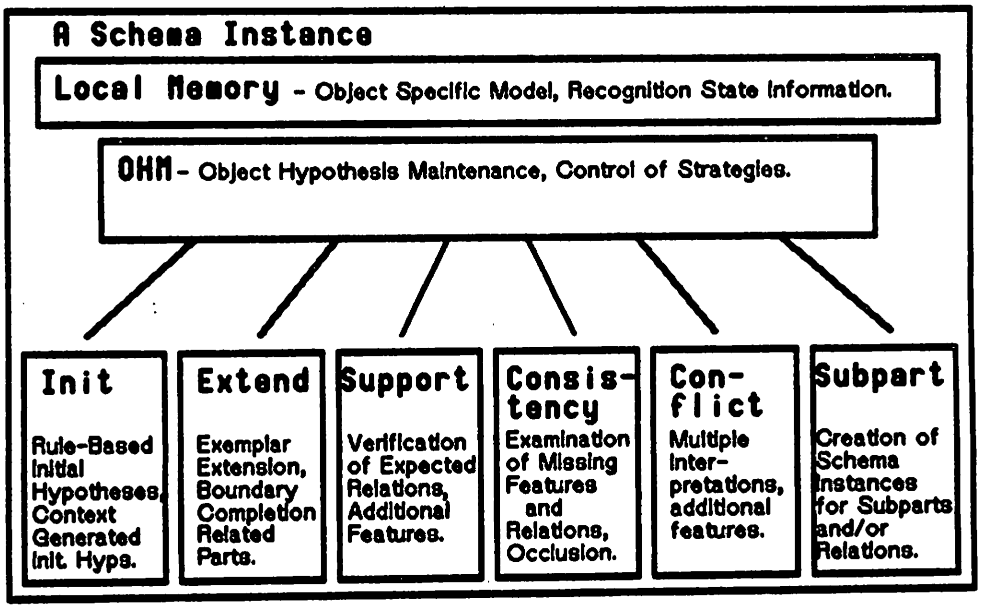

A Schema Instance
Local Memory - Object Specific Model, Recognition State Information.
OHM - Object Hypothesis Maintenance, Control of Strategies.
Init
Rule-Based Initial Hypotheses, Context Generated Init. Hyps.
Extend
Exemplar Extension, Boundary Completion Related Parts.
Support
Verification of Expected Relations, Additional Features.
Consistency
Examination of Missing Features and Relations, Occlusion.
Conflict
Multiple Interpretations, additional features.
Subpart
Creation of Schema Instances for Subparts and/or Relations.

Figure 9: A Schema template – the OHM.

symbols road-color, road-texture, centerline-found, left-line-found, right-line-found, gravel-found, and side-structure-found (the last refers to caution signs, stop signs and telephone poles). The schema uses an object specific routine to map the endorsements of the internal hypothesis onto the confidence values of the external hypothesis. This preserves modularity by allowing one schema to inspect another's external hypothesis without knowing the details of its construction. While it is too early to draw any firm conclusions about this mechanism for combining evidence, its performance in this experiment has been encouraging.

The internal structure of each schema was organized according to a schema template (Figure 9). This divided the recognition process into seven subtasks, each handled by a different strategy. The OHM, or *O*bject *H*ypothesis *M*aintenance strategy, provides the interface to other instances. It creates and maintains the external object hypothesis, and reads the external hypotheses of other instances from the blackboard. The Initial Hypothesis strategy provides a focus of attention mechanism. It generates inexpensive but plausible internal hypotheses for other strategies to corroborate or deny. The other strategies are Extension, which extends the current hypothesis based on partial results; Support, which exploits redundancy by seeking additional evidence for hypotheses; Negative Information, which must reason about any internal failures or unexplained data; Conflict, which handles inconsistencies with the external hypotheses or other instances; and Subpart, which implements the calling hierarchy discussed above.

Certain strategy types were more successful than others in this experiment. No negative information strategy has yet been implemented. The extend and support strategy types were both heavily used, but appear to fulfill a similar function; no single schema uses both. It is also necessary that different schema's conflict strategy be mutually consistent; to this end, a single, generic conflict strategy was developed and used for all objects. The OHM strategies, although individualized for each schema, were all very similar. Their code could be replaced by a simpler, declarative format.

3.2 Examples of Schema Behavior

At some level, the behavior of a heuristically based program is best described on a case by case basis. This section shows examples of how different objects are recognized in the Schema System. Included are interpretations of both man-made (e.g. road, caution sign) and natural objects. We also discuss areas in which the system needs to be extended.

3.2.1 Recognizing Road

Road is recognized through the interactions of the *road* and *roadline* schemas, with *gravel*, *caution sign*, *stop sign* and *telephone pole* also available to provide support. The initial hypothesis strategy for road uses an initial hypothesis generation tool similar to that discussed in [Lehr87] known as the Rulesys. The initial hypotheses generated for three images are given in Figure 10. In general, road sections near the observer are found. However, in Figure 10c the system also puts forth the sky as a road hypothesis, and many more distant road sections are usually missed, as is the case in 10a and 10b. The regions indicated by the Rulesys are grouped by the initial hypothesis strategy, which causes the OHM to activate the road subpart strategy. This in turn activates roadline schemas focussed on any long, high contrast lines that overlap the road hypothesis. It also invokes a schema to look for gravel (of the type often found on roadsides) along the edges of its hypotheses.

The interplay between the road and roadline schemas is quite involved. The roadline schema (like all schemas) attempts to find support for its hypothesis. One pair of endorsements which fosters belief is 1) feature space support from the Rulesys, and 2) a bounding set of parallel lines. These two endorsements, coinciding with a partially supported road hypothesis, will lead the roadline schema instance to have confidence **belief* in its external hypothesis. It will also have **belief* in a hypothesis that has not been supported by the Rulesys if the corresponding road hypothesis is strongly believed. The item most likely to foster belief in a road hypothesis, however, is a believed roadline in the appropriate position (identified as either centerline, left sideline or right sideline). Hence finding the first segment of roadline increases the system's belief in other complementary roadline hypotheses. Figure 11 shows how these factors interact in an actual image.

Once road and roadline have been identified, their hypotheses can generally be improved. While sections of roadline near the camera tend to be demarcated by strong parallel lines, the quality of these lines deteriorates as the road recedes. Often, the line extractor will lose portions of one or both lines. In addition, projections of roads in images often bend, either because the road turns or because the actual ground plane is not flat. This inevitably leads to broken lines. However, once a section of a roadline has been identified, this knowledge can be used as an *island of certainty* to extend the roadline hypothesis past the point where the line structure breaks down. This in turn allows the hypothesis for road to be extended, since the relationship between the road and roadline is established (centerline, right sideline or left sideline). Figure 12 shows the road hypotheses before expansion.

or deny. The other strategies are Extension, which extends the current hypothesis based on partial results; Support, which exploits redundancy by seeking additional evidence for hypotheses; Negative Information, which must reason about any internal failures or unexplained data; Conflict, which handles inconsistencies with the external hypotheses or other instances; and Subpart, which implements the calling hierarchy discussed above.

Certain strategy types were more successful than others in this experiment. No negative information strategy has yet been implemented. The extend and support strategy types were both heavily used, but appear to fulfill a similar function; no single schema uses both. It is also necessary that different schema's conflict strategy be mutually consistent; to this end, a single, generic conflict strategy was developed and used for all objects. The OHM strategies, although individualized for each schema, were all very similar. Their code could be replaced by a simpler, declarative format.

3.2 Examples of Schema Behavior

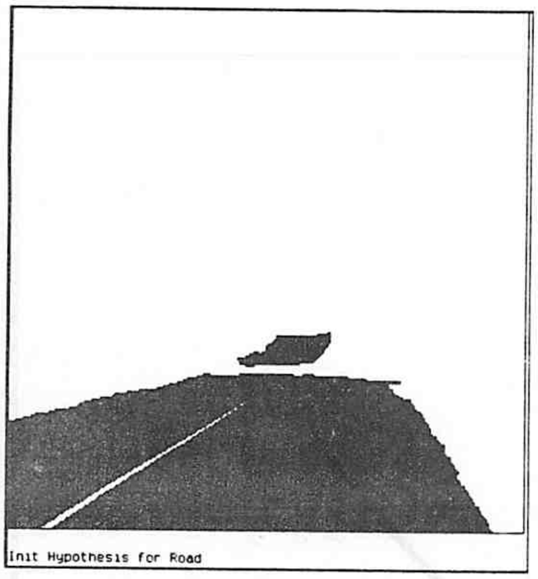
At some level, the behavior of a heuristically based program is best described on a case by case basis. This section shows examples of how different objects are recognized in the Schema System. Included are interpretations of both man-made (e.g. road, caution sign) and natural objects. We also discuss areas in which the system needs to be extended.

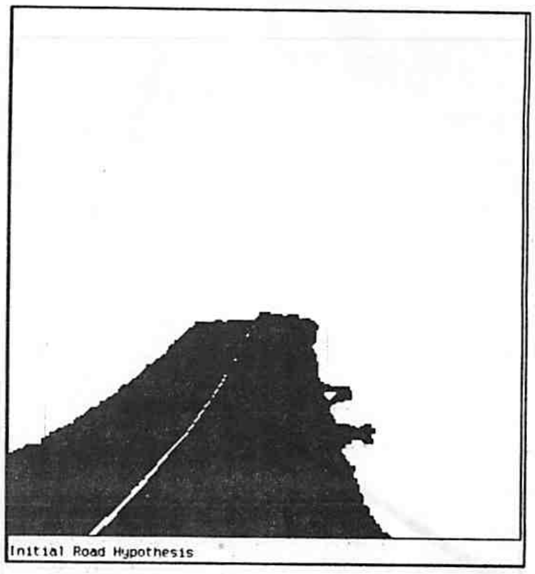
3.2.1 Recognizing Road

Road is recognized through the interactions of the *road* and *roadline* schemas, with *gravel*, *caution sign*, *stop sign* and *telephone pole* also available to provide support. The initial hypothesis strategy for road uses an initial hypothesis generation tool similar to that discussed in [Lehr87] known as the Rulesys. The initial hypotheses generated for three images are given in Figure 10. In general, road sections near the observer are found. However, in Figure 10c the system also puts forth the sky as a road hypothesis, and many more distant road sections are usually missed, as is the case in 10a and 10b. The regions indicated by the Rulesys are grouped by the initial hypothesis strategy, which causes the OHM to activate the road subpart strategy. This in turn activates roadline schemas focussed on any long, high contrast lines that overlap the road hypothesis. It also invokes a schema to look for gravel (of the type often found on roadsides) along the edges of its hypotheses.

The interplay between the road and roadline schemas is quite involved. The roadline schema (like all schemas) attempts to find support for its hypothesis. One pair of endorsements which fosters belief is 1) feature space support from the Rulesys, and 2) a bounding set of parallel lines. These two endorsements, coinciding with a partially supported road hypothesis, will lead the roadline schema instance to have confidence **belief* in its external hypothesis. It will also have **belief* in a hypothesis that has not been supported by the Rulesys if the corresponding road hypothesis is strongly believed. The item most likely to foster belief in a road hypothesis, however, is a believed roadline in the appropriate position (identified as either centerline, left sideline or right sideline). Hence finding the first segment of roadline increases the system's belief in other complementary roadline hypotheses. Figure 11 shows how these factors interact in an actual image.

Once road and roadline have been identified, their hypotheses can generally be improved. While sections of roadline near the camera tend to be demarcated by strong parallel lines, the quality of these lines deteriorates as the road recedes. Often, the line extractor will lose portions of one or both lines. In addition, projections of roads in images often bend, either because the road turns or because the actual ground plane is not flat. This inevitably leads to broken lines. However, once a section of a roadline has been identified, this knowledge can be used as an *island of certainty* to extend the roadline hypothesis past the point where the line structure breaks down. This in turn allows the hypothesis for road to be extended, since the relationship between the road and roadline is established (centerline, right sideline or left sideline). Figure 12 shows the road hypotheses before and after expansion.





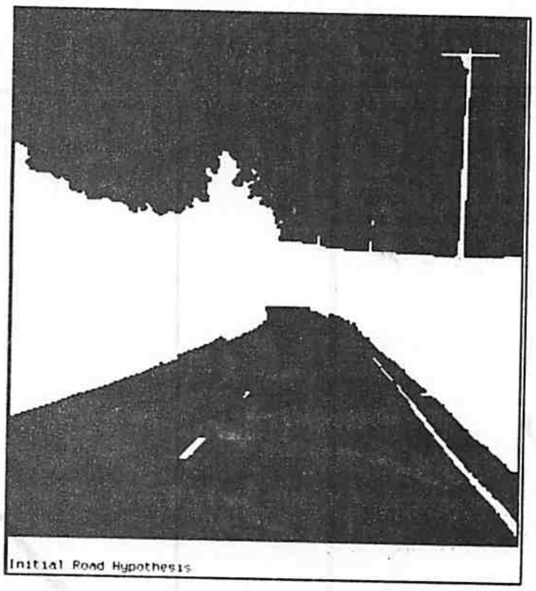


Figure 10a,b,c. Initial road hypotheses. (a) Image in Figure 2. (b) Image in Figure 3. (c) Image in Figure 4.

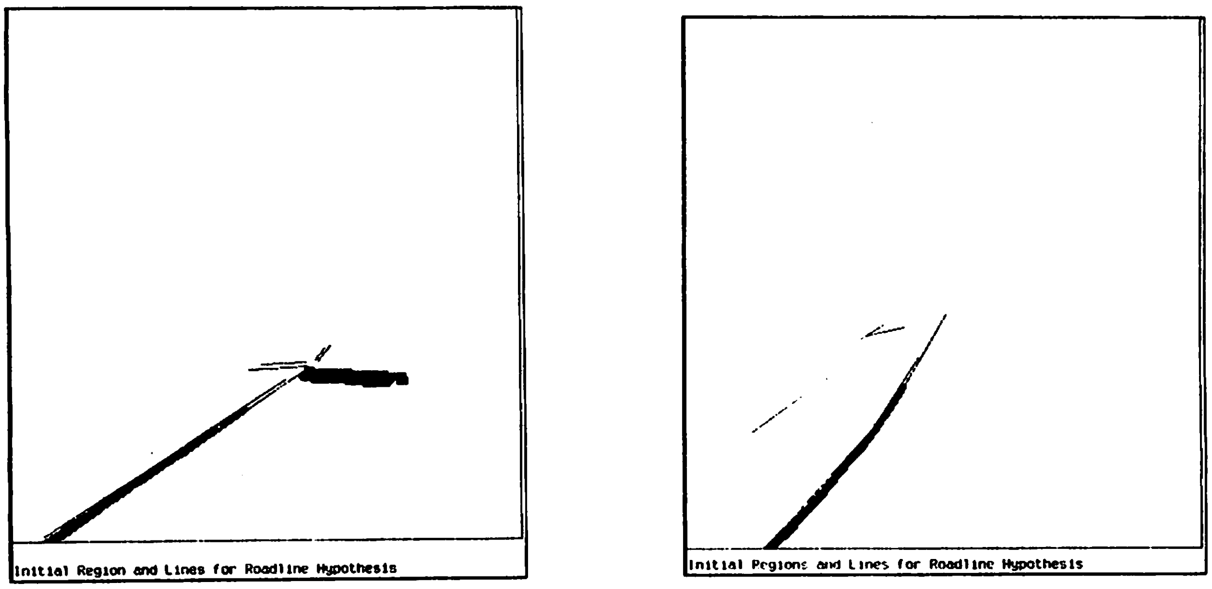

Initial Region and Lines for Roadline Hypothesis
Initial Regions and Lines for Roadline Hypothesis

Figure 11a,b. Interaction of lines and regions in initial roadline hypotheses.

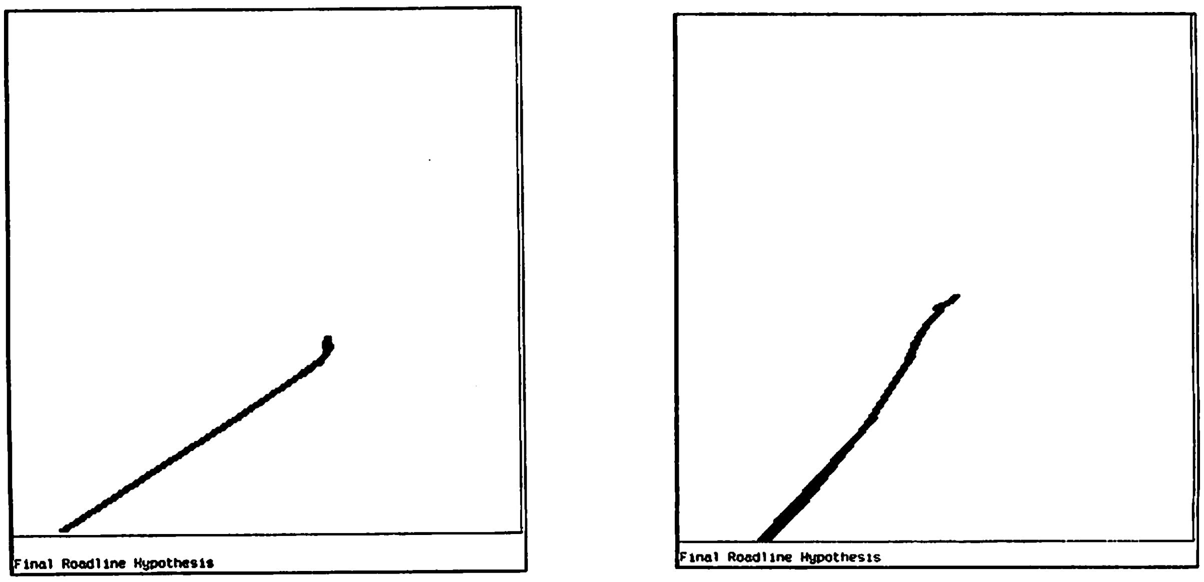


Figure 11c,d. Results of roadline expansion.

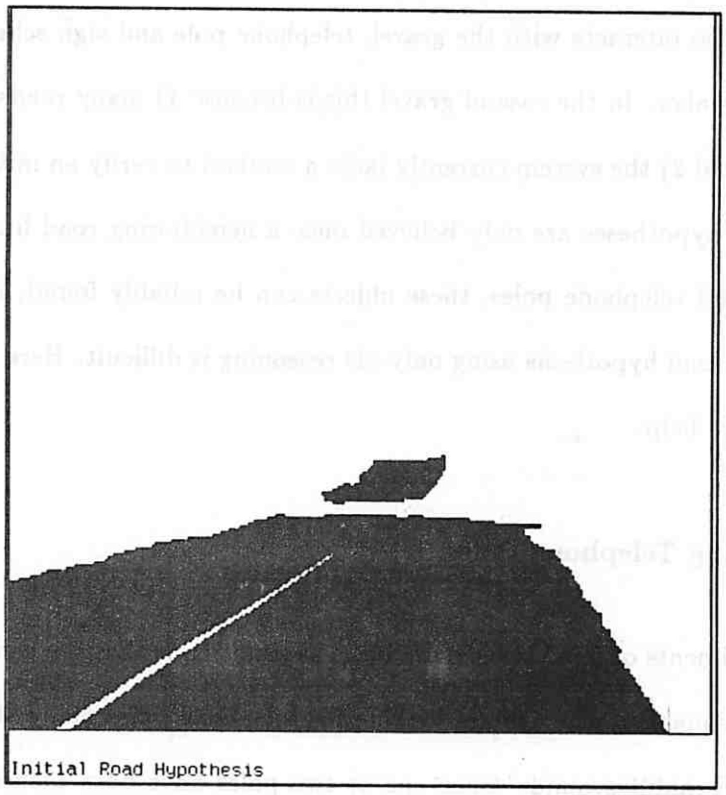
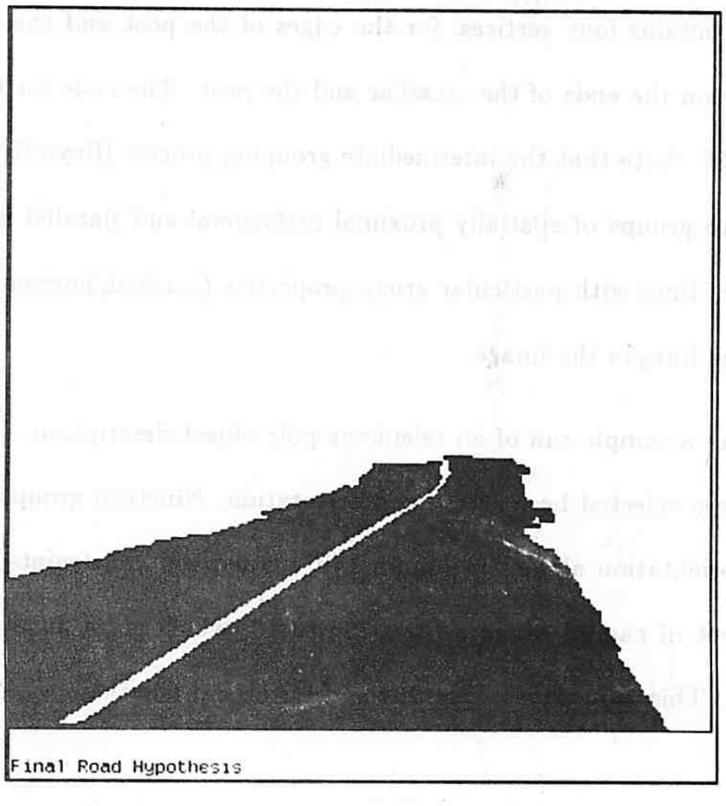


Figure 12a,b. Road hypotheses before and after expansion. (a) Initial region and line hypotheses for Figure 2. (b) Final hypothesis.



The road schema also interacts with the gravel, telephone pole and sign schemas. Here, however, the interaction is weaker. In the case of gravel this is because 1) many roads in the world are not lined with gravel, and 2) the system currently lacks a method to verify an initial gravel hypothesis. As a result, gravel hypotheses are only believed once a neighboring road has been confirmed. In the case of signs and telephone poles, these objects can be reliably found, but establishing their relationship to the road hypothesis using only 2D reasoning is difficult. Here again, future work in 3D reasoning should help.

3.2.2 Recognizing Telephone Poles

In this set of experiments on road scenes, the basic assumption is that the camera is in the position of a vehicle. Consequently, schemas are looking for telephone poles along the side of the road in the foreground and middleground. Once one or two poles have been identified, perspective and relational information can be used to hypothesize the location and size of additional poles. The 2d model for a pole is a simple 'T' made by an upright and a crossbar. The object description for such a model contains four vertices, for the edges of the post and the crossbar. The model ignores short edges on the ends of the crossbar and the post. The code for the object description is given in Figure 13. Note that the intermediate grouping process [Reyn87] has already collected the image lines into groups of spatially proximal orthogonal and parallel lines. So OVARC can search for groups of lines with particular group properties (vertical/horizontal, proximal) instead of processing all the lines in the image.

Let us now consider a sample run of an telephone pole object description. In Figure 14a, a set of candidates have been selected by location and orientation. Nineteen groups out of a possible 323 were selected by orientation alone. In Figure 14b, on-demand constraints have been computed, paring down the set of candidate lines from 266 to 10, which gives an average of 2.5 lines per description vertex. This reduction is due to the vertical and length constraints on the post lines.

```
(defvar *phone-pole-description*
(define-object-description
  '((upright-right-edge
     ((spo_related crossbar-bottom-edge
                   crossbar-top-edge)
      (spp_related upright-left-edge)
      (vertical upright-right-edge)
      ((length$>$ 20) upright-right-edge)
      (right-of upright-left-edge)))
    (upright-left-edge
     ((spo_related crossbar-top-edge
                   crossbar-bottom-edge)
      (spp_related upright-right-edge)
      (vertical upright-left-edge)
      ((length$>$ 20) upright-left-edge)))
    (crossbar-top-edge
     (((atop .25) upright-right-edge
                  upright-left-edge)
      (spo_related upright-right-edge)
      (spo_related upright-left-edge)
      (spp_related crossbar-bottom-edge)
      ((atop 0) crossbar-bottom-edge)))
    (crossbar-bottom-edge
     (((atop .25) upright-right-edge
                  upright-left-edge)
      (spo_related upright-right-edge)
      (spo_related upright-left-edge)
      (spp_related crossbar-top-edge)))
   )))
```

Figure 13. Telephone pole object description. In the code the name of each description node is followed by an association list in which the first item of each association pair is the name of a constraint and the second item is a list of nodes, each of which is related to the description node by the named constraint. The constraints spo_related and spp_related are for spatially proximate orthogonal and parallel pairs of lines. (atop .25) is a relation in which one line falls somewhere in the top one-quarter of another line A constraint label like (atop .25) has a function name as the first element; the other elements allow the constraint to be parameterized by supplying extra arguments to the function. The (length> 20) restricts objects of interest to be in the foreground.

Since verticality and size are the strongest constraints in the description, selection of candidates for the post reduces the amount of computation necessary to eliminate candidates for the crossbar edges.

Figure 14c shows the output of the subgraph isomorphism computation. There were no false positives for this image. The foreground telephone pole is easily matched because of the strength of its lines and the regularity of its relations with respect to the object description. The success in finding the middle-ground pole is due the strength of the line-extraction process [Weis86] and the effectiveness of the grouping methodology [Reyn87].

Once the object has been found, the model can be completed by the telephone pole schema instance. Knowledge about size, symmetry and positional relations can be used to fill in the edge information, confirm region identification and contribute support to hypotheses about road location and direction, and ground plane irregularities.

3.2.3 Recognizing Road Signs

Road signs are in general easily found by their bright colors. These initial color-based hypotheses can then be verified by locating the supporting post. Since the hypothesis determines the location and size of the post, finding it is an inexpensive task. The Schema System misses one stop sign. The image in Figure 15 contains two distinct stop signs; one that is large and close to the viewer, and another that is farther away and at an oblique angle. The nearby sign is easily and correctly found. The other is proposed as an initial hypothesis, but cannot be confirmed because the linear structure of the post is missing. This is a case where knowledge directed control of the low level processes is called for. The system knows where to focus its attention; but although the capability for extracting the post through a top-down line extraction routine is being developed, it has not yet been integrated into the Schema System [Kohl87a][Kohl87b]. Figureamroad1-results shows the

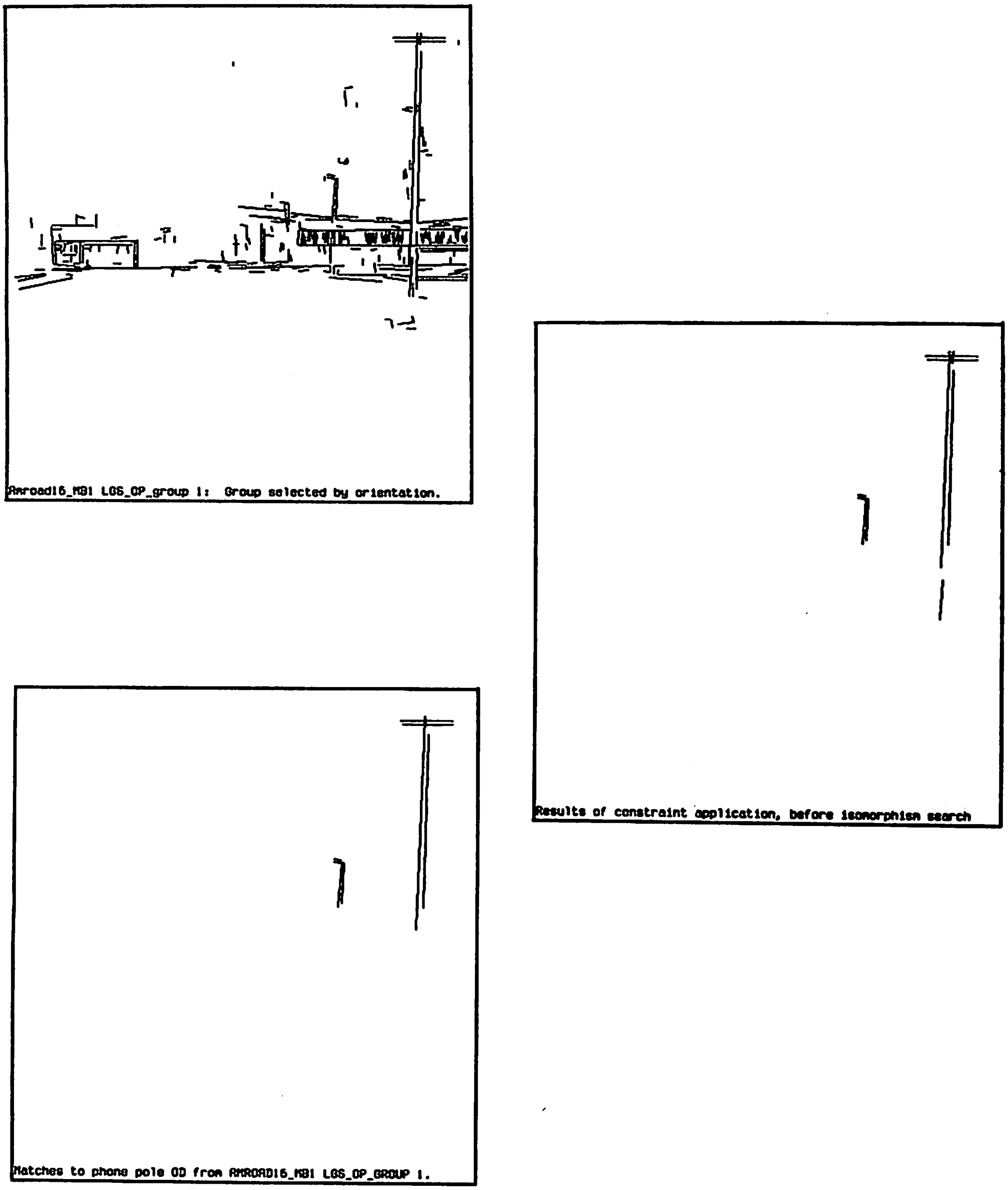

Amroad16_MB1 LGS_OP_group 1: Group selected by orientation.
Results of constraint application, before isomorphism search
Matches to phone pole OD from AMROAD16_MB1 LGS_OP_GROUP 1.

Figure 14a,b,c. Stages in telephone pole object description match.

initial stop sign hypotheses and the single, final believed stop sign hypothesis.

3.2.4 Recognizing Sky

The initial hypothesis strategy for sky is configured as a generator. It initiates a few internal hypotheses on the basis of color and then monitors their progress. If at any point these hypotheses have all been refuted, the strategy relaxes its criteria and generates a new set of hypotheses. This continues until a strong hypothesis for sky is found or the strategy runs out of reasonable candidates. An example of this can be seen in Figure 17.

3.3 Experimental Environment

The Schema System runs on a TI Explorer lisp machine. The ISR uses a VAX 11/750 (connected to the Explorer by a Chaosnet) as a host for its database. Once the image has been segmented and its lines extracted and grouped, the interpretation process takes about 50 minutes. Much of this time is spent communicating with the VAX over the chaosnet; the simulation of parallelism also slows down the system. A subjective estimate is that an efficient, production implementation of the system could interpret an image in under 10 minutes.

4 Conclusion

A knowledge based computer vision system that produces effective image interpretations on complex natural scenes has been developed. There are several directions of research being pursued with this system. This first is to port it onto a newly acquired multiprocessor. Second, the knowledge engineering tools for implementing schemas in a new domain need to be extended. The goal of this research is for a small development team to be able to implement a set of schemas of the

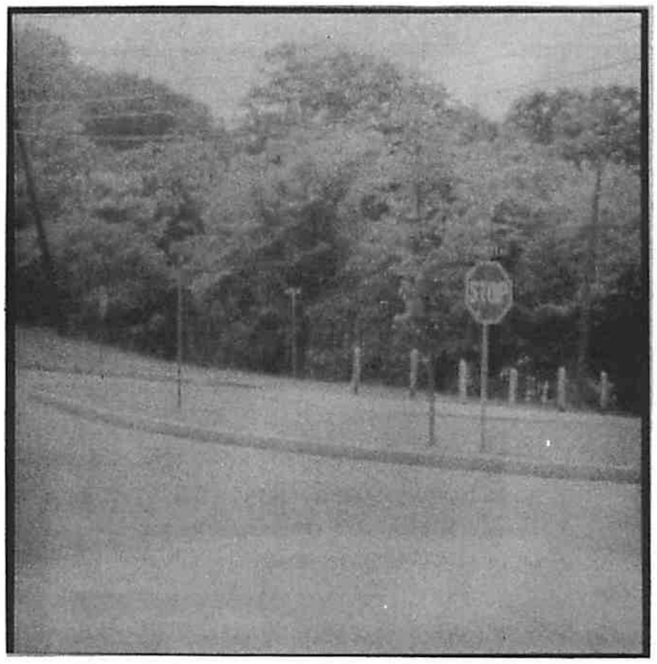


Figure 15: Photograph of intersection with stop signs.

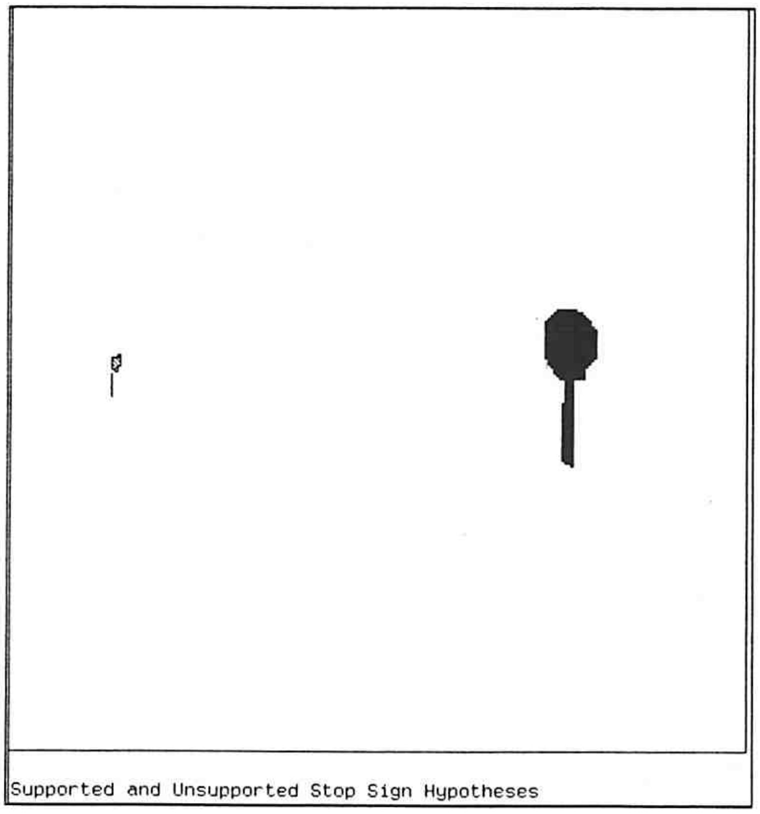


Figure 16. One supported (black) and one unsupported (grey) stop sign hypothesis.

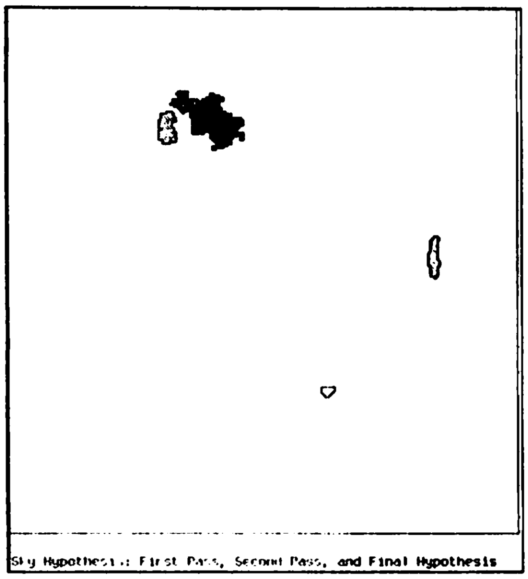

Sky Hypothesis: First Pass, Second Pass, and Final Hypothesis

Figure 17. Initial sky hypothesis (clear); relaxed hypothesis (grey and black); final hypothesis (black).

complexity shown here in a week, rather than the month required for this effort. We intend to test this capability in the future by building a knowledge base for an aerial image domain. Third, the current knowledge bases for house scenes and road scenes need to be further developed and generalized, including object schemas which are sensitive to image resolution and object distance. Most importantly, the system must be expanded to use three-dimensional reasoning to create three-dimensional object hypotheses.

5 Acknowledgements

The writers wish to thank a number of people for their help: George Reynolds and Ross Beveridge for rectilinear line grouping, Rob Belknap and Rick Latty for rule development, Charlie Kohl for knowledge-directed token extraction, Nancy Lehrer for automatic rule acquisition, Michael Boldt for line extraction, Rich Weiss, Kelly Murray and Arnie Rosenberg for advice on the object verification knowledge source.

References

[Arbi78] M.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.

[Arbi81] M.A. Arbib, "Perceptual structures and distributed motor control", in ***Handbook of Physiology: the nervous system, II Motor control***, V.B. Brooks (Ed.), Amer. Physiol. Soc., Bethesda, MD., pp. 1449-1480.

[Belk85] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied To Complex Aggregations of Image Events," ***Proc. DARPA IU Workshop***, Miami Beach, FL, December 1985.

[Drap86] B. Draper, A. Hanson, and E. Riseman, "A Software Environment for High Level Vision," COINS Technical Report, University of Massachusetts at Amherst, 1986, [in preparation].

[Erma80] L.D. Erman, F. Hayes-Roth, V.R. Lesser and D.R. Reddy. "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," ***Computing Surveys***, 12(2) (June 1980), pp. 213-253.

[Hans78] A. Hanson and E. Riseman, "VISIONS: A Computer System for Interpreting Scenes", in ***Computer Vision Systems***, Hanson & Riseman, eds., Academic Press, N.Y., 1978, pp. 303-333.

[Hewi77] C. Hewitt, "Viewing Control Structures as Patterns of Passing Messages," ***Artificial Intelligence***, Vol. 8, #3 (June 1977), pp. 323-364.

[Kohl87a] C. Kohl, Ph.D. Dissertation, Deparment of Computer and Information Science, University of Massachusetts at Amherst, in preparation, 1987.

[Kohl87b] C. Kohl, A. Hanson, and E. Riseman, "Goal-Directed Control of Low-Level Processes for Image Interpretation," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Lehr87] N. Lehrer, G. Reynolds, and J. Griffith, "Initial Hypothesis Formation in Image Understanding Using an Automatically Generated Knowledge Base," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Reyn87] G. Reynolds and J.R. Beveridge, "Searching for Geometric Structure in Images of Natural Scenes," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Shaf86] S.A. Shafer, A. Stentz and C.E. Thorpe, "An Architecture for Sensor Fusion in a Mobile Robot," in *CMU Blackboard Workshop Proceedings,* Pittsburgh, 1986.

[Ullm76] J.R. Ullman, "An algorithm for subgraph isomorphism," *Journal Assoc. Comput. Mach.,* Vol. 23, 1976, pp. 31-42.

[Weis86] R. Weiss and M. Boldt, "Geometric Grouping Applied to Straight Lines", *Proceedings on the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Miami, FL, June, 1986, pp. 489-495.

[Weym86] T.E. Weymouth, "Using Object Descriptions in a Schema Network For Machine Vision," Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst. Also, COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.