Learning Blackboard-based Scheduling Algorithms for Computer Vision

Bruce A. Draper Allen R. Hanson Edward M. Riseman
Dept. of Computer Science
University of Massachusetts
Amherst, MA., USA. 01003*

Abstract

The goal of image understanding by computer is to identify objects in visual images and (if necessary) to determine their location and orientation. Objects are identified by comparing data extracted from images to an *a priori* description of the object or object class in memory. It is a generally accepted premise that, in many domains, the timely and appropriate use of knowledge can substantially reduce the complexity of matching image data to object descriptions. Because of the variety and scope of knowledge relevant to different object classes, contexts and viewing conditions, blackboard architectures are well suited to the task of selecting and applying the relevant knowledge to each situation as it is encountered.

This paper reviews ten years of work on the UMass VISIONS system ([12, 13]) and its blackboard-based high-level component, the schema system ([6, 7]). The schema system could interpret complex natural scenes when given carefully crafted knowledge bases describing the domain, but its application in practice was limited by the problem of model (knowledge base) acquisition. Experience with the schema system convinced us that learning techniques must be embedded in vision systems of the future to reduce or eliminate the knowledge engineering aspects of system construction.

The Schema Learning System (SLS) is a supervised learning system for acquiring knowledge-directed object recognition (control) strategies from training images. The recognition strategies are precompiled reactive sequences of knowledge source invocations that replace the dynamic scheduler found in most blackboard systems. Each strategy is specialized to recognize instances of a specific object class within a specific context. Since the strategies are learned automatically, the knowledge base contains only general-purpose knowledge sources rather than problem-specific control heuristics or sequencing information.

*This work was supported by DARPA under TACOM contract DAAE07-91-C-R035, RADC under contract F30602-91-C-0037 and the NSF under grant CDA-8922572.

1 Introduction

The goal of computer image understanding is to identify objects in visual images and (if necessary) to determine their location and orientation (i.e. their pose). Objects are identified by comparing data extracted from images to an *a priori* description of the object or object class in memory. It is a generally accepted premise that, in many domains, the timely and appropriate use of knowledge can substantially reduce the complexity of matching image descriptions to object models.

The blackboard architecture was first proposed by researchers working on automatic speech recognition ([11]). In order to interpret one-dimensional acoustic signals in real time it was necessary to integrate knowledge at different levels of abstraction and to focus computational resources on the most promising hypotheses first. The solution was the blackboard architecture, in which knowledge is encapsulated in independent procedural modules called knowledge sources (KSs). Knowledge sources exchange information about hypotheses through a central blackboard, which serves both to buffer data and to insulate KSs from each other. A heuristic scheduler associated with the central blackboard decides which knowledge source should be executed on each cycle, invoking only those KSs which involve promising hypotheses. Reviews of blackboard technology can be found in [20] and [10].

Faced with many of the same problems that arise during speech recognition, some vision researchers adopted the blackboard model. The VISIONS system used a blackboard model to recognize common 2D views of 3D objects ([12]). Nagao and Matsuyama designed an aerial photograph interpretation system around the blackboard model, with knowledge sources for reasoning about the size, shape, brightness, location, color and texture of a region ([19]). PSEIKI ([1]) combined the blackboard programming model with the Shafer-Dempster theory of evidence to recognize 2D objects. Some of the difficulties and advantages of using blackboards for vision are discussed in [6].

2 Summary of the VISIONS Schema System

From the beginning, the goals of the schema system were different from those motivating the designs of most other computer vision systems. First, the schema system was designed to recognize both man-made and natural objects in outdoor scenes. As a result, it could not make many of the assumptions used by other model-based systems, such as assuming polygonal objects. Instead, the schema system had to use many types of information about object classes, including shape, color, texture and context, and to opportunistically select the appropriate model features to compare to the image data; for this reason, the schema system was initially conceptualized as a blackboard system ([12]).

Second, the schema system was meant to recognize not just single objects but entire scenes. The premise was that it is easier to recognize objects in context than in isolation, and that a partial interpretation generated for one region of an image can produce constraints on the interpretation of the rest of the image. This led to a distributed blackboard system in which both knowledge and computation were partitioned at a coarse-grained semantic level. Coarse-grained knowledge was encapsulated in the schemas, with each schema serving as a specialized "expert" at recognizing a single object class.

A schema instance is invoked for each object class hypothesized to be in the image data. These instances execute independent (potentially concurrent) processes called recognition strategies and communicate asynchronously through a global blackboard. The control component of each schema schedules the application of general purpose procedures, called knowledge sources, to gather the "right kind" of support for (or against) its hypothesis. Competition and cooperation among the schema instances results in the combination of multiple, independent "object experts" into a large scale system which constructs internally consistent interpretations.

2.1 Components of the Schema System

The schema system consists of five basic components: the schema hierarchy, the blackboard, the knowledge sources, the interpretation (control) strategies, and mechanisms for evidence representation and combination. Each of these is discussed very briefly in the following sections; more detail may be found in [7].

2.1.1 The Schema Hierarchy

The schema system partitions both knowledge and computation in terms of natural object classes for a given domain. Schemas reside in class and part/subpart hierarchies; each class of objects and object parts has a corresponding schema which stores all object and control knowledge specific to the recognition of instances of that class. Knowledge about expected object contexts and relationships to other objects is represented in the system by extending the concept of an object to include contextual or scene configurations; as objects, these entities also have schemas. A subcontext or "sub-scene" is like an object part; it is related to its parent scene or context in predicatable ways.

2.1.2 Knowledge Sources

Knowledge sources are general-purpose procedures that generate the levels of abstract image descriptions required in an image understanding system. Knowledge sources span the gamut of traditional techniques in image processing (e.g. region, line, curve, and surface extraction, feature measurement, etc), through intermediate level processes such as initial object hypothesis generation and grouping operations to generally useful tools and techniques such as graph matching. The compile-time arguments and parameters supplied to a general-purpose knowledge source as part of the recognition strategy may specialize it for a particular purpose.

KSs typically create, manipulate, and construct abstract symbolic representations of image events stored symbolically in the ISR ([4]), a database specially designed for image understanding systems. The database supports associative retrieval, spatial relations, and multi-level representations, and has been optimized for spatial retrieval. In the current version of the schema system, which has recently been extended to include three-dimensional object representations and three- dimensional interpretation, over 40 KSs are available as basic building blocks.

2.1.3 Interpretation Strategies

Interpretation strategies, or simply strategies, are control programs that run within each schema and fulfill the role of a scheduler. Strategies are a procedural encoding of knowledge about which knowledge sources to apply and in what order to apply them. To make maximal use of parallelism, schemas may have multiple concurrent strategies corresponding to different methods for recognizing an object or to different conditions under which recognition must take place. Schemas can also contain strategies for different subtasks, such as initial hypothesis generation and hypothesis verification, as well as for managing the internal bookkeeping details of the schema, such as updating the global blackboard when necessary and detecting and resolving conflicts related to the hypothesis.

Each schema instance acquires information pertinent to the hypothesis it is pursuing. Some of this information is generic, to the extent that its semantics are not object dependent. For example, the degree of confidence in a hypothesis, as well as its (2D) image location and (3D) world location, is generic information, because every object hypothesis has a confidence level and an image location, and most have a meaningful 3D location. The generic information about an object hypothesis is recorded in a global hypothesis.

Most of the information acquired by a schema instance, on the other hand, is object specific. Information about how well an image region matches an expected color, for example, is non-generic since its importance depends on the object model. A color match may be important for finding trees, but less so for recognizing automobiles. For this reason, all of the information about which KSs support a particular hypothesis and which do not is considered private to the schema instance, and is not included in the global hypothesis.

2.1.4 Blackboard Communication

The schema system is built around a global blackboard (Figure 1). The global hypotheses written to the blackboard represent the image interpretation as it evolves. Schemas communicate with each other by writing to and reading from the blackboard, dynamically exchanging information about their respective hypotheses. Although the blackboard is divided into sections corresponding to the object classes (rather than processing levels, as in other systems [20]), schemas may read and write freely over the entire blackboard. The division into sections gives some assurance that a schema will not have to search through a large number of irrelevant messages. At the same time, each schema instance maintains its own local blackboard for recording private information (Figure 2).

The distinction between the global and local blackboards was motivated both by computational and knowledge engineering concerns. Computationally, most of the information generated by an interpretation strategy concerns which KSs have been run, what each KS returned, etc. While this information is crucially important within the schema instance for making dynamic control decisions, it is of little importance to other schema instances. If the strategies associated with multiple concurrent schema instances continually dump this

Figure 1: The schema system's global blackboard (from [7]).

Figure 2: A local blackboard and its relationship to the global blackboard of the schema system (from [7]).

information to the global blackboard and then read it back again, the blackboard quickly becomes a computational bottleneck. The local blackboards alleviate this problem by reducing the message traffic on the global blackboard.

From a knowledge engineering viewpoint, the distinction between the global and local blackboards promotes modularity by allowing only the strict "global hypothesis" protocol to be exchanged between schemas. Each schema can maintain local information in an idiosyncratic manner on its local blackboard, allowing the schema designer the freedom of any appropriate knowledge representation and control style. At the same time, because schemas communicate with each other only through global hypotheses, the designer of a new schema is assured of a smooth join to the remainder of the system.

Local blackboards are also partitioned into sections, where each section usually corresponds to a level of abstraction. The local blackboard is accessible to all the strategies within a particular schema instance, but only those strategies. As a result, while messages to the global blackboard are required to conform to a strict protocol, local blackboard messages can be highly schema specific.

2.1.5 Evidence Accumulation

The current version of the schema system takes a particularly simple view of evidence representation and combination. Confidence values lie along a coarse, five point ordinal scale: 'no evidence', 'slim-evidence', 'partial-support', 'belief', and 'strong-belief'. When combining evidence, a heuristic mechanism is used that involves the specification of key pieces of evidence that are required to post an object hypothesis with a given confidence to the global blackboard. Subsets of secondary evidence are used to raise or lower these confidences. Specifications of these subsets, and the effect their confidence has on the overall confidence, is part of the knowledge engineering effort involved in constructing a schema. Although this method of evidence representation and accumulation may lack considerably from a theoretical point of view, it has worked surprisingly well in interpretation experiments on images of New England house and road scenes [7].

2.2 Knowledge Engineering in the Schema System

Schemas are assembled by specifying (1) the appropriate set of knowledge sources to be used, (2) a set of strategies which conditionally sequence their application, and (3) a function to translate internal evidence into a confidence in the global hypothesis. One of the main impediments to wide scale experimentation with the schema system has been the time and energy required to design a schema. Schema construction can be viewed as an exercise in experimental engineering, in which prototype schemas are developed using existing system resources. These schemas must then be tested on a representative set of objects/images, failures noted and analyzed, and the schemas re-engineered to account for the failures. In many cases, the descriptive information provided by the knowledge sources may be inadequate. In this case, new knowledge sources must be developed and tested (often a major research effort in its own right), integrated into the system, and the schemas re-engineered to make use of the new information.

The problem of knowledge base construction has been a focus of research for several years. In artificial intelligence, researchers have focused on how to extract knowledge from experts, a scenario which does not apply to computer vision. Vision researchers have concentrated instead on how knowledge bases are specified. By restricting the message types written to the global blackboard, the schema system enforces schema modularity in an attempt to make schemas easier to declare and improve. The SPAM project at CMU developed a high-level language for describing objects ([18]). Work in Japan has involved both automatic programming efforts and higher-level languages for specifying image operations ([17]).

Despite these efforts, however, there are no blackboard-based vision systems in operation today that are capable of recognizing more than a couple dozen objects, and we conclude from their absence that the knowledge engineering problem has not yet been satisfactorily solved.

3 Learning Scheduling Strategies

For the last two years we have taken a different approach to knowledge base development. Instead of making the knowledge base easier to program, we have decided to take the programmer out of the loop. Our goal is a knowledge-directed vision system that learns its own interpretation strategies.

As a first step in this direction we have designed the Schema Learning System (SLS; [8]), as shown in Figure 3. SLS's task is to learn interpretation strategies for object classes. In particular, it learns object recognition strategies that minimize the cost of achieving a *recognition goal*, specified by a level of representation and accuracy parameters. For example, a recognition goal might be to recognize the location and orientation of the building represented as a 6D coordinate transformation, accurate to within five percent of the distance from the building to the camera. Alternatively, a simpler goal would be to recognize the image location of a building in terms of its centroid and accurate to within two pixels.

SLS learns recognition strategies from training images and their "solutions", where the solutions are in the form of the recognition goal. Thus if the goal is to recognize the pose of an object, the pose of the object in each training image must be known; alternatively, if the goal is to recognize the image position of an object, then the position of the object in the training images must be known. In general, SLS learns to generate hypotheses that match the solutions provided for the training images.

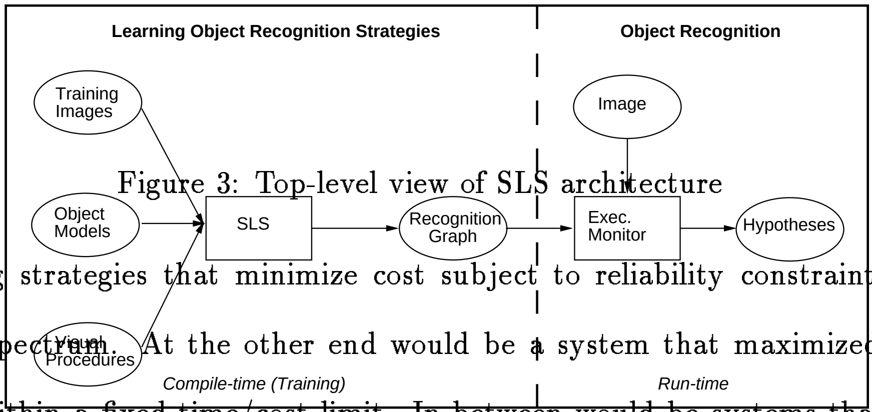

Figure 3: Top-level view of SLS architecture

In learning strategies that minimize cost subject to reliability constraints, SLS is at one end of a spectrum. At the other end would be a system that maximized recognition performance within a fixed time/cost limit. In between would be systems that maximized a utility function weighing cost against robustness. In the applications we have focused on, however – navigation and robotic assembly – safety comes first, leading us to design a system that minimizes cost subject to hard robustness constraints, rather than the other way around.

As implied by Figure 3, SLS's operations can be divided into two parts: a compile-time (or "learning-time") component in which SLS develops recognition strategies, and a run-time component in which the interpretation strategies are applied to new images. In general, SLS has been designed to optimize run-time performance, at the expense of compile-time (learning) efficiency.

The learning task is made easier by two simplifying assumptions. First, SLS learns to recognize instances of each object class independently. This is easier than learning concurrent, cooperating strategies. Second, SLS is given a set of knowledge sources (KSs) from which to build its recognition strategies. Thus SLS is not required to learn new KSs, but rather to learn to schedule KSs and combine evidence within the blackboard paradigm.

3.1 Modeling the Interpretation Process

SLS adopts the blackboard processing model in that interpretation is viewed as a process of applying knowledge sources to hypotheses. Hypotheses are proposed statements about the image or its interpretation, whose type is determined by their *level of abstraction.* Common levels of abstraction for computer vision include: image, region, (2D) image line segment, line segment group, (3D) world line segment, (3D) orientation vector, surface patch, face, volume, object and context. An "interpretation" is a set of believed hypotheses at the level of abstraction specified by the recognition goal (also called the *goal level*). Knowledge sources are procedures from the image understanding literature (e.g. region segmentation, line extraction or vanishing point analysis) that can be applied to one or more hypotheses.

SLS, however, refines the blackboard model of interpretation by constraining knowledge sources to fall into one of two classes. *Generation knowledge sources* (GKSs) create new hypotheses or transform old hypotheses from one level of abstraction to another. For example, region segmentation is a GKS, since it creates new hypotheses (regions) when applied to the image; a stereo line matching algorithm, which transforms two (2D) image line segments into a new (3D) world line segment, is another GKS. *Verification knowledge sources* (VKSs), on the other hand, return discrete feature values describing the hypotheses they are applied to. An example of a VKS is a pattern matching algorithm that determines the degree to which the color or texture of an image region matches the expected color or texture of the object. In general, any routine which measures features of hypotheses can be converted into a VKS by discretizing its results.

3.2 Recognition Graphs

Interpretation strategies are represented in SLS as generalized multi-level decision trees called *recognition graphs* that direct both hypothesis formation and hypothesis verification, as shown in Figure 4. The premise behind the formalism is that object recognition is a series of small verification tasks interleaved with representational transformations. Recognition begins with trying to verify hypotheses at a low level of abstraction, separating to the extent possible hypotheses that are reliable from those that are not. Verified hypotheses (or at least, hypotheses that have not been rejected) are then transformed to higher levels of abstraction, where new verification processes take place. The cycle of verification followed by transformation continues until hypotheses are verified at the goal level, or until all hypotheses have been rejected.

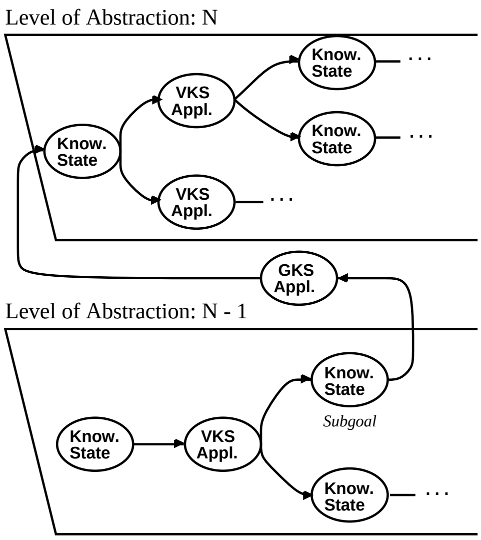

Figure 4: A recognition graph. Levels of the graph are decision trees that verify hypotheses using verification knowledge sources (VKSs). Hypotheses that reach a subgoal are transformed to the next level of abstraction by a generation knowledge source (GKS).

The structure of the recognition graph reflects the verification/transformation cycle. Each level of the recognition graph is a decision tree that controls hypothesis verification at one level of abstraction by invoking VKSs to gather support for or against each hypothesis. When sufficient evidence is accumulated for a hypothesis, a GKS transforms it to another level of abstraction, where the process repeats itself.

As defined in the field of operations research (e.g. [14], Chapter 15), decision trees are a form of state-space representation composed of alternating *choice states* and *chance states.* When searching for a path from the start state to a goal state, an agent at a choice state is allowed to choose which chance state to go to next. At chance states, the next choice states are chosen probabilistically[1]. The search process is therefore similar to using a game tree against a probabilistic opponent.

In SLS, the choice states are hypothesis knowledge states as represented by sets of hypothesis feature values. The choice to be made at each knowledge state is which VKS (if any) to execute next. Chance states in the tree represent VKS applications, where the chance is on which value the VKS will return. Hypothesis verification is an alternating cycle in which the control strategy selects which VKS to invoke next (i.e., which feature to compute), and the VKS probabilistically returns a feature value. Thus hypotheses advance from knowledge states to VKS application states and then on to new knowledge states. The cycle continues until a hypothesis reaches a subgoal (verification) state, indicating that it should be transformed to a higher level of abstraction, or a failure state, indicating that it is unreliable and should be rejected.

The goal is for SLS to learn in advance what VKS to choose at each knowledge state and to build a recognition graph with just one option at each choice node, thereby eliminating the need for run-time control decisions. Sometimes, however, the readiness of a VKS to be executed cannot be determined until run-time, in which case SLS will leave several options

[1]Operations research terminology is based on trees rather than spaces, so it refers to choice nodes and chance nodes rather than choice states and chance states, and to leaf nodes and root nodes rather than goal states and start states.

at a choice node, sorted in order of desirability[2]. At run-time the system will choose the highest-ranking VKS that is ready to be executed.

3.3 The Schema Learning System (SLS)

The Schema Learning System (SLS) constructs interpretation strategies represented as recognition graphs. SLS is given (1) a set of knowledge sources; (2) a recognition goal; and (3) a set of training images with solutions. It builds recognition strategies that minimize the expected cost of satisfying the recognition goal by a three-step process: *exploration* applies KSs to the training images in order to build a statistical characterization of each KS and to generate examples of correct and incorrect hypotheses; *learning from examples* determines how hypotheses are generated by learning which GKSs to use from the examples generated during exploration; and finally *graph optimization* optimizes hypothesis verification by selecting the order VKSs are applied in at each level of abstraction.

3.3.1 Exploration

The exploration algorithm exhaustively applies the KSs in the knowledge base to the training images, beginning with those KSs (both VKSs and GKSs) that can be applied directly to images. Some of these GKSs will produce more abstract hypotheses such as regions or lines, and KSs are applied to these hypotheses to produce still more hypotheses, until eventually every KS has been applied to every possible hypothesis for each training image. (Since the correct solution is known for every training image, exploration can be made more efficient by abandoning false hypotheses at low levels of abstraction, at the risk of biasing the learning algorithm.)

[2]This is just one of many complications that arise from multiple-argument knowledge sources. In general, we will describe SLS as if all KSs took just one argument in order to keep the description brief; see [9] for a more complete description.

The purpose of exploration is twofold. First, it generates examples of how correct hypotheses (i.e. hypotheses that match the user's solutions) are generated from images, for use by the learning from examples algorithm. Second, it provides a basis for characterizing KSs. In particular, the expected cost $C(k|n)$ of every KS k is estimated, where n is a set of feature values. In addition, it also estimates the likelihood of each feature value e, $P(e|k,n)$, for every VKS. In both cases, if feature combination n is seen too rarely during training to provide reliable estimates, the values of $C(k)$ and $P(e|k)$ are substituted for $C(k|n)$ and $P(e|k,n)$, respectively.

3.3.2 Learning from Examples (LFE)

SLS's second step looks at the correct interpretations produced during exploration and infers from them a scheme for generating good hypotheses while minimizing the number of false hypotheses; this is achieved by tracing the sequence of GKSs used to produce each good hypotheses. For example, a correct 3D pose hypothesis might be generated by fitting a plane to a set of 3D line segments. If so, the pose hypothesis is dependent on the plane fitting GKS. It is also dependent on whatever GKS created the 3D line segments, and any GKSs needed to create its arguments, etc. The result of tracing back a hypothesis' dependencies is an AND/OR tree like the one shown in Figure 5. 'AND' nodes in the tree result from GKSs that require multiple arguments, such as stereo matching. 'OR' nodes in the tree occur when a hypothesis is redundantly generated by more than one GKS (or a single GKS applied to multiple sets of hypotheses).

Each dependency tree is an example of how correct hypotheses are generated, and the example is generalized by replacing the hypotheses in the tree with their feature vectors. In other words, Figure 5 should be interpreted as showing how correct poses can be created by applying certain GKSs to less abstract hypotheses with specific sets of features, where the

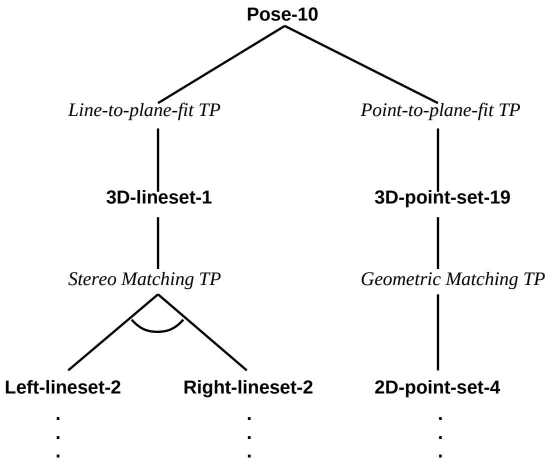

Figure 5: An example of a dependency tree showing the different ways that one correct pose hypothesis can be created during training. AND nodes (shown with an arc) signify that both branches must be satisfied, while OR nodes (shown without an arc) signify that at least one branch must be satisfied.

features of the hypotheses are viewed as preconditions for the GKS. In the example shown in Figure 5, correct pose hypotheses may be created by applying either the line-to-plane-fit GKS to line hypotheses with the feature values of 3D-lineset-1 or the pt-to-plane-fit GKS to point hypotheses with the feature values of 3D-point-set-19.

By definition, any set of GKSs and GKS preconditions that satisfies a dependency tree would generate the correct hypothesis the tree represents. A set of GKSs and GKS preconditions that satisfies the dependency trees of all the correct hypotheses will generate correct hypotheses for every training image. SLS finds such a set of preconditioned GKSs by ANDing the dependency trees together and converting the resulting expression to conjunctive normal form (DNF). Every conjunctive subterm of the DNF expression is a set of preconditioned GKSs that will generate a correct hypothesis for every training image, and SLS selects the subterm that generates the fewest false hypotheses along with the correct ones.

The AND/OR dependency tree is converted into DNF by a standard algorithm that first converts its subtrees to DNF and then either merges the subterms (if the root is an OR node) or takes the symbolic cross-product of the subterms (if the root is an AND node). SLS, however, is designed to find just the minimal term of the DNF expression; as a result, whenever one subterm of a DNF is a logical superset of another term, the superset term can be pruned from the expression.

3.3.3 Optimization

As was stated earlier, recognition graphs interleave verification and transformation, using VKSs to gather evidence to verify or reject hypotheses, and GKSs to transform them to higher levels of abstraction. In the previous step, the LFE algorithm not only learned which GKSs to use, it also learned what conditions (feature values) a hypothesis must have before a GKS transforms it to the next level. The optimization algorithm builds decision trees at each level of abstraction that minimize the expected cost of acquiring these features; hypotheses not meeting the preconditions are rejected. Decision trees are built at each level by laying out a graph of all possible sequences of knowledge states and VKS applications and then pruning it to leave just the tree that minimizes the expected cost.

At each level of abstraction, graph layout begins with a start state, representing a hypothesis for which no features have been computed. The start state is a choice state, and the choice is which feature to compute first. Chance states are generates for every VKS that can be applied to a hypothesis in the start state, and these chance states lead to new choice states, depending on which discrete value the VKS returns. These choice states lead to still more chance states, and so on, until a choice state are reached that either satisfy the preconditions of a GKS (i.e. a subgoal state), or is incompatible with the preconditions (i.e. a failure state).

Once the graph has been created it is pruned to where each choice state contains just a single option. The pruning requires a single pass through the graph, starting at the subgoal and failure nodes and working toward the start state. At each chance (VKS application) state, the expected cost of reaching a subgoal or failure node from the application state is calculated. At each choice state, the chance state with the lowest expected cost is selected and all others are removed from the graph. (In the event that the optimal VKS might not be executable at run time, it sorts the remaining VKSs in order of least to greatest expected cost rather than removing them.

More formally, the subgoal states and failure states at one level of a recognition graph are the *terminal* states for that level. The cost of promoting a hypothesis from choice state n to a terminal state is the Expected Decision Cost (EDC) of state n, and the expected cost of reaching a terminal state from state n through chance state (VKS) k is the Expected Path Cost (EPC) of n and k. We refer to the possible discrete outcomes of a VKS k as $R(k)$, and the probability of a particular value e being returned as $P(e|k,n), e \in R(k)$.

The EDC's of knowledge states can be calculated starting with the terminal states and working backwards through the recognition graph. Clearly, the EDC of a subgoal or failure state is zero:

$$EDC(n) = 0, \quad n \in \{terminal\ states\}.$$

The expected path cost of reaching a terminal state from a chance state is:

$$EPC(n,k) = C(k) + \sum_{e \in R(k)} (P(e|n,k) \times EDC(n \cup e))$$

where n is the previous choice state expressed as a set of feature values, $n \cup e$ is the knowledge state that results from VKS k returning feature value e and $C(k)$ is the estimated cost of applying k.

The EDC of a choice state, then, is the smallest EPC of the VKSs that can be executed at that state:

$$EDC(n) = \min_{k \in KS(n)} (EPC(n, k))$$

where $KS(n)$ is the set of VKSs applicable at node n.

The equations above establish a mutually recursive definition of the expected decision cost of a choice state. The EDC of a choice state is the EPC of the optimal VKS application at the state; the EPC of a chance state is the expected cost of the VKS plus the remaining EDC after the VKS has been applied. The recursion bottoms out at terminal nodes, whose EDC is zero. Since every path through the object recognition graph ends at either a subgoal or a failure node, the recursion is well defined. Furthermore, since the EDC of a level's start state estimates the expected cost of verifying a hypothesis at that level of abstraction, the EDCs of all the start states can be combined with estimates of the number of hypotheses generated at each level to estimate the expected run-time of the strategy as a whole.

3.4 Experimental Results

The previous sections give a simplified description of a complex system that has only recently been implemented. Because the system is new, complete and thorough experiments testing its success both as a knowledge engineering tool and as a machine learning system are only now underway; in this section we report the results of one such experiment.

The goal of the experiment was to test SLS within the scenario of learning to accurately recognize the pose of a complex object from an approximately known viewpoint; other experiments are testing its ability to perform 2D recognition and to perform 3D recognition from arbitrary viewpoints. The motivation for testing recognition from an approximately known viewpoint first is that the images are more self-similar, and therefore require smaller training sets. Experiments on recognition from an unknown viewpoint require considerably larger training sets than used here.

The training images for the current experiment were a set of twenty images of the Marcus Engineering building on the UMass campus, including the ones shown in Figures 6 and 7, taken along a dirt path at distances ranging from three to four hundred feet from the building. The pictures were taken level to gravity (i.e. with zero tilt and roll) but with small rotations (pan) from one image to the next, leaving four degrees of freedom in the pose of the building: three for location and one for rotation. The training solutions (i.e., ground truth) were determined by matching hand-selected points in each image to points on a wire-frame model of the building, and using Kumar's algorithm to determine the pose of the building ([16]). The goal was to learn a strategy that could recognize the pose of the building to within 10° rotation (pan), 5% depth (scale) and 1° of the correct image angle (the angle between the optical axis and a ray from the focal point to the object).

The knowledge sources available to SLS included a geometric matcher for comparing wire-frame models to image data ([2]), perspective analysis routines for estimating orientations ([5, 15]), a line grouping system ([21]), a pattern classifier ([3]), and a template matching routine. It was also given knowledge sources for checking domain constraints such as distance from an object to the camera or the height of an object above (or below) the camera plane.

SLS was tested by a "leave one out" scheme in which strategies were trained on nineteen images and tested on the twentieth; the error between the best-verified (highest confidence) hypothesis for an image and the user's solution is shown in Table 1. (When the highest confidence value was shared by multiple hypotheses, Table 1 shows the average of their errors.) SLS's strategies generated pose hypotheses that matched the user's solutions (to within the accuracy parameters in the recognition goal) in nineteen of the twenty tests. As an example, Figure 8 shows the pose found for image twenty (shown in Figure 6). In two

Figure 6: The first Marcus Engineering image

Figure 7: The last Marcus Engineering image

of the tests, however, poses with excessive error were rated more highly than the correct hypotheses, so the recognition strategy succeeded in only seventeen of twenty trials.

A system's failures are often more informative than its successes. In this case, one of the most efficient strategies for finding the pose of the building depends on finding the trihedral junction corresponding to the building's upper left hand corner. (The junction indicates both the building's image location and, through perspective distortion, its orientation, specifying three of four degrees of freedom.) This strategy works on nineteen of the twenty images in the training set; in image 6, however, the top line of the building is missing, perhaps due to noise, and no trihedral junction can be found. Consequently, when SLS is trained on the nineteen images other than image 6, it notices that every training sample produces a trihedral junction and concludes that no backup strategy is necessary. It then fails to find the building in image 6. Interestingly, when image 6 is included in the training set, SLS notices that the trihedral junction method can fail and includes otherwise redundant techniques based on vanishing points.

The result on image 6 raises the question of how to proceed when SLS's recognition strategy fails. Traditional blackboard systems, when confronted with a failure, will switch to backup recognition strategies. The same approach is possible with SLS, since each conjunctive subterm of LFE's DNF expression is the basis for a recognition strategy. In general, however, there may be no way to detect a failure, since a negative result may correctly indicate that the object is not in the image. Moreover, there is no way to know which of the alternate strategies might be effective, nor is there a guarantee that any of them will work. In most situations, the most effective strategy is to take another picture and try the first strategy again.

SLS's strategy also failed to verify the best hypotheses for images 7 and 9. Goal-level hypothesis verification is a classification task, and these failures are indicative of the difficulty of hypothesis classification in general, and of classification by discrete features in particular. In this case, none of the available VKSs are sensitive enough to detect small changes in a pose's rotation or scale. The verified hypothesis for image 9, for example, was within one degree (pan) of being within tolerance, a distinction not captured by the coarsely-discretized features used here. Increasing the number of discrete values per feature can reduce this type of error, but there will alwaus be some distinctions that are beyond the resolution of any set of discrete features. Finely-discretized features also require larger training sets, since the relative impact of each feature value must be estimated.

In addition to being robust, SLSs strategies are also supposed to be efficient. Unfortunately, they cannot be directly compared to hand-crafted strategies, since no such strategies are available for this domain and knowledge base. In can be noted, however, that the exhaustive interpretations of the training images by the exploration algorithm took, on average, $3\frac{1}{2}$ hours per image on a TI Explorer lisp machine, while the strategies learned by SLS recognized the building in an average of 310.6 seconds, or about $2\frac{1}{2}\%$ of the time required by exhaustive search. (All times were measured as the sum of KS execution times, and do not include system overhead.)

4 Conclusion

It is generally accepted that the timely and appropriate use of relevant knowledge can substantially reduce the search encountered in matching image data to object descriptions. This premise is supported by our experience with the schema system, a blackboard-based object recognition system that embedded object-specific knowledge in multiple concurrent processes called schemas. Unfortunately, the problem of how to acquire and structure knowledge has limited the schema system as well as other knowledge-based vision systems to highly con-

	1	2	3	4	5	6	7	8	9	10
Prob	0.79	0.87	0.88	0.86	0.86	–	0.90	0.89	0.87	0.89
Pan(°)	4.95	2.50	7.87	1.75	6.21	–	2.58	2.95	11.00	4.48
Scale(%)	2.26	1.21	0.45	1.43	4.44	–	11.23	0.60	4.87	2.24
ImAngle(°)	0.13	0.03	0.07	0.14	0.13	–	0.21	0.09	0.21	0.25

	11	12	13	14	15	16	17	18	19	20
Prob	0.88	0.88	0.86	0.86	0.86	0.85	0.86	0.97	0.90	0.90
Pan(°)	3.27	1.65	3.58	0.07	1.16	6.95	6.14	1.40	1.33	0.82
Scale(%)	4.83	1.39	1.35	2.33	0.97	0.73	2.47	2.58	3.44	1.34
ImAngle(°)	0.08	0.04	0.20	0.08	0.04	0.15	0.10	0.13	0.12	0.05

Table 1: The errors between the most probable pose hypothesis and the true pose for each test image. Pan refers to difference in rotation about the gravitational axis measured in degrees, scale to the distance from the camera to the object measured as a percentage of the true distance, and image angle to the angle between a ray from the camera to the object and the camera's optical axis. The user's tolerance thresholds were 5% scale, 10° pan and 1° image angle.

strained domains. We feel that if blackboard-based vision is to become practical, recognition strategies will have to be acquired automatically.

The schema learning system (SLS) is an experimental system for learning recognition strategies from training images. The eventual goal is to completely eliminate the knowledge engineering task; at the moment, a human is still required to supply sets of knowledge sources (GKSs and VKSs) and training images. The previously time-consuming process of supplying control knowledge, however, has been replaced by SLS.

SLS's strategies have two advantages besides ease of construction over hand-crafted ones. First, they are robust to the extent that they will correctly interpret every training image. Hand-crafted strategies may or may not have this property. Second, the are efficient in that they minimize the number of (goal-level and intermediate-level) hypotheses generated when recognizing an object, and minimize the cost of verifying or rejecting hypotheses at each level of abstraction. As a result, they are generally optimal in the sense of minimizing the total expected cost of recognition, although counterexamples are theoretically possible.

Figure 8: The most probable pose generated for the image shown in Figure 6 (Image #1 in Table 1).

As with all learning systems, however, SLS's strategies may fail if the test image is significantly different from all of the training images (e.g., image 6). Our current efforts in extending SLS focus on developing a theoretical bounds on the reliability of strategies on test images based on the history of their training. We are also interested in developing an adaptive form of SLS's batch-oriented learning algorithm for use on an autonomous vehicle.

References

[1] K.M. Andress and A.C. Kak. "Evidence Accumulation and Flow of Control in a Hierarchical Reasoning System," *AI Magazine* 9(2):75–94 (Summer 1988).

[2] J. Ross Beveridge, Richard Weiss and Edward M. Riseman. "Optimization of 2-Dimensional Model Matching," *Proc. of the DARPA Image Understanding Workshop,* Palo Alto, CA., June 1989. Morgan-Kaufman Publishers, Los Altos, CA. pp. 815-830.

[3] Carla Brodley and Paul Utgoff. "Linear Machine Decision Trees," submitted to *Machine Learning.*

[4] John Brolio, Bruce A. Draper, J. Ross Beveridge and Allen R. Hanson. "The ISR: an intermediate-level database for computer vision", *Computer*, 22(12):22-30 (Dec. 1989).

[5] Robert T. Collins and Richard S. Weiss. "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere." *Proc. of the International Conference on Computer Vision,* Osaka, Japan, 1990.

[6] Bruce A. Draper, Robert T. Collins, John Brolio, Allen R. Hanson and Edward M. Riseman. "Issues in the Development of a Blackboard-Based Schema System for Image Understanding," in [10] pp. 189-218.

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

[8] Bruce A. Draper and Edward M. Riseman. "Learning 3D Object Recognition Strategies", *3rd International Conference on Computer Vision*, Osaka, Japan, Dec., 1990. pp. 320-324.

[9] B.A. Draper. *Learning Object Recognition Strategies,* forthcoming Ph.D. dissertation, Univ. of Massachusetts.

[10] Robert Engelmore and Tony Morgan. *Blackboard Systems.* Addison-Wesley, London, 1988.

[11] 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,* 12:349–383 (1980).

[12] Allen R. Hanson and Edward M. Riseman. "VISIONS: A Computer System for Interpreting Scenes," in *Computer Vision Systems,* Hanson and Riseman (eds.), Academic Press, N.Y., 1978. Pp. 303-333.

[13] Allen R. Hanson and Edward M. Riseman, "The VISIONS image understanding system – 1986" in *Advances in Computer Vision,* C. Brown (ed.), Erlbaum: Hillsdale, N.J., 1987.

[14] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* Holden-Day, Inc: San Francisco, 1980.

[15] Ken-ichi Kanatani. "Constraints on Length and Angle," *Computer Vision, Graphics, and Image Processing,* 41:28-42 (1988).

[16] Rakesh Kumar and Allen R. Hanson. "Model Extension and Refinement using Pose Recovery Techniques," *Journal of Robotics Systems,* 9(6), 1992.

[17] Takashi Matsuyama. "Expert Systems for Image Processing: Knowledge-Based Composition of Image Analysis Processes" *Computer Vision, Graphics, and Image Processing,* 48:22–49 (1989).

[18] David M. McKeown, Jr., Wilson A. Harvey, and Lambert E. Wixson. "Automating Knowledge Acquisition for Aerial Image Interpretation" *Computer Vision, Graphics, and Image Processing,* 46:37–81 (1989).

[19] Makoto Nagao and Takashi Matsuyama. *A Structural Analysis of Complex Aerial Photographs.* Plenum Press, N.Y., 1980.

[20] H. Penny Nii. "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures," *AI Magazine,* 7(2):38-53 (1986).

[21] George Reynolds and J. Ross Beveridge. "Searching for Geometric Structure in Images of Natural Scenes," *Proc. of the DARPA Image Understanding Workshop,* Los Angeles, CA., Feb. 1987. pp. 257-271 (Morgan-Kaufman Publishers, Inc.).