Learning 3D Object Recognition Strategies

Bruce A. Draper Edward M. Riseman

Computer & Information Science Department,
University of Massachusetts at Amherst *

Abstract

Knowledge-directed vision uses object descriptions to guide the visual recognition process. It is potentially efficient, since objects can be found by searching for only their most distinctive characteristics. For example, roadside warning signs in the U.S. can be identified by their distinctive yellow color at the same time office buildings are located by their linear structure. Similarly, the 3D position of some objects can be recovered through vanishing point analysis, while the location of other objects is determined by perspective-based point matching.

A major problem in knowledge-directed vision is the construction of object-directed control strategies. Existing knowledge-directed systems rely on user-supplied heuristics to guide the recognition process. Specifying these heuristics has proven a difficult and time-consuming process. Worse still, there is no guarantee that the resulting strategies are either effective or optimal.

This paper considers the problem of automatically learning knowledge-directed control strategies. In particular, it addresses the problem of learning object-specific recognition strategies from object descriptions and sets of interpreted training images. A separate recognition strategy is developed for every object in the domain. The goal of each recognition strategy is to identify any and all instances of the object in an image, and give the 3D position (relative to the camera) of each instance. The goal of the learning process is to build a strategy that minimizes the expected cost of recognition, subject to accuracy constraints imposed by the user.

1 Introduction

The goal of computer vision is to identify objects in an image or image sequence, and to locate them in three dimensions. For example, when a picture of a house is presented to a vision system it should be able to determine both the type of object (i.e. house) and its 3D position relative to the camera. Both object identification and object positioning are equally important, and we refer to their combination as *3D object recognition.*

In order to recognize an object in an image, a system must compare data extracted from the image to a description of the object class. This description, in turn, can suggest what features to look for in the image. For example, a description in memory of "house" might constrain the shape of a house, but not its color, since houses can be painted almost any color. Therefore, when looking for houses, shape primitives should be extracted, but not color features. The best strategy for recognizing a house (or any object) is determined by its properties.

This paper considers the problem of automatically learning object-specific control strategies from object descriptions and sets of interpreted training images. A separate recognition strategy is developed for every object to be recognized. The goal of each recognition strategy is to identify any and all instances of the object in an image, and give the 3D position (relative to the camera) of each instance. The goal of the learning process is to build a strategy that minimizes the expected cost of recognition, subject to accuracy constraints imposed by the user.

In this work, object recognition is modeled as a process of applying visual knowledge sources to hypotheses. Knowledge sources are routines from the image understanding literature, such as 2D→3D point matching, vanishing point analysis and straight line extraction. Hypotheses are intermediate-level statements about the image and/or 3D world, and can occur at many levels of abstraction. Examples of hypotheses include straight line segments, 3D orientation vectors and volumes. At each step in the recognition process, a knowledge source is applied to one or more hypotheses. The result is either a new hypothesis or a discrete evidence value reflecting the quality of the original hypotheses.

Recognition strategies are represented by recognition graphs, which are similar in many ways to decision trees. Unlike decision trees, however, recognition graphs direct hypothesis creation as well as hypothesis classification or verification. Object-specific strategies are learned in a two step process. The first step involves learning which hypotheses should be generated. The second step learns how to verify them efficiently.

This work extends the knowledge-based approach by replacing the ad-hoc control heuristics of other systems with well-motivated control and classification decisions. The user, instead of supplying heuristics in the form of if-then rules or confidence functions, specifies accuracy requirements. The system selects KSs that minimize the expected cost of recognition while achieving the specified accuracy.

2 System Overview

The top-level design of the Schema Learning System (SLS) is shown in Figure 1. SLS infers object recognition strategies from an object description and a set of training images. The object description contains parameterized knowledge sources that the user feels may help to recognize the object. SLS tests each KS in the object description by applying it the training images and comparing the results to the user's interpretation. In this way, SLS empirically tests both the accuracy and the efficiency of the KSs. SLS uses this data to construct a recognition graph. At

*This work was supported in part by DARPA and RADC under contract number F30602-87-C-0140, by DARPA and U.S. Army ETL under contract number DACA76-89-C-0017, and by the National Science Foundation under grant number DCR-8500332.

CH2934-8/90/0000/0320$01.00 © 1990 IEEE

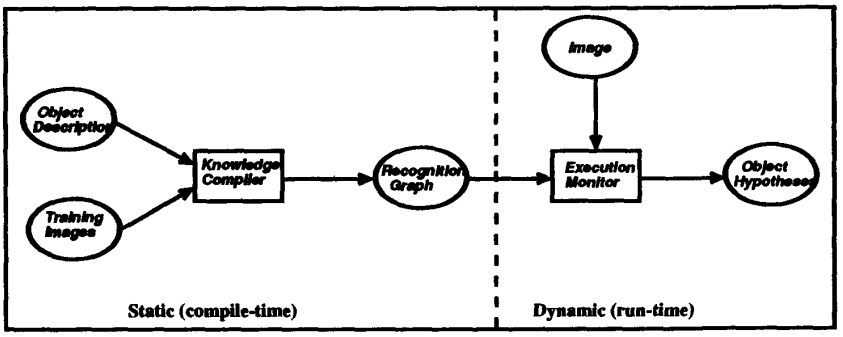

Figure 1: The Schema Learning System (SLS)

run-time, this graph is used to efficiently and accurately recognize the object in images.

The heart of the learning problem is the box labeled "knowledge compiler" in Figure 1. It reasons about competing object recognition strategies using the recognition graph formalism described below. To support this formalism, it distinguishes between two types of knowledge sources. *Hypothesis generation* KSs create new hypotheses. Vanishing point analysis, which creates a 3D orientation vector from 2D line segment is one example. *Verification* KSs supply symbolic endorsements about previously existing hypotheses. Pattern classification algorithms that compare a region's color to the expected color of an object are examples; such algorithms are modified to return symbolic endorsements such as "good-match","average-match", etc.

2.1 Recognition Graphs

Recognition graphs are most easily thought of as generalized decision trees with *choice nodes* and *chance nodes* (see [1], chapter 15). Choice nodes in the graph correspond to hypothesis knowledge states; the choice to be made is which verification KS should be executed next. When each choice node in a graph has just one option, then the recognition graph represents a particular strategy. When the choice nodes present multiple options, the recognition graph represents several possible strategies, corresponding to the different options of processing at that point. Chance nodes in the graph represent the run-time application of a verification KS. The branches from a chance node represent the different possible outcomes of the KS at run-time, in terms of the evidential endorsements it might return.

Figure 2 shows part of a recognition graph. Hypotheses begin at start states (to the left in Figure 2) and advance to the right through a series of knowledge states (depicted as parallelograms). Each knowledge state is a set of endorsements. These endorsements represent what the system has learned about the hypothesis so far. Hypotheses are advanced from one state to another by verification KSs (shown as ellipses with an inscribed "V"). Verification KSs return endorsements which describe the hypothesis.

Although conceptually similar to decision trees, recognition graphs are a quite distinct representation. As their name implies, recognition graphs are not trees but graphs. When two verification KSs are applied to a hypothesis, the hypothesis ends up in the same knowledge state no matter what order the knowledge sources are applied in. Hence there can be two different paths through the recognition graph to a single state, making it a directed acyclic graph (DAG) rather than a tree.

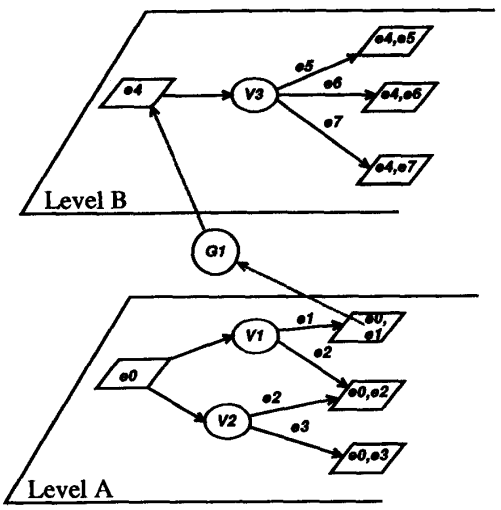

Figure 2: Part of a Recognition Graph

Furthermore, because computer vision requires reasoning over many levels of abstraction, recognition graphs are divided into layers, one for each type of hypothesis to be considered. Thus a typical recognition graph might have levels of data abstraction for regions, 2D lines, 2D points, 3D orientation vectors, planes, and 3D poses. At each level in the graph different verification KSs are defined. For example, a KS that tests the color of a hypothesis would be used at the region level, but not the 3D orientation level.

Image interpretation proceeds across levels of abstraction through generation knowledge sources (shown as ellipses with inscribed "G"s in Figure 2). Generation KSs link knowledge states at one level of abstraction to knowledge states at another, show how new hypotheses are created. For example, a vanishing point analysis KS that makes a 3D orientation vector from a set

of 2D line segments is represented as a link from a knowledge state at the line-set level to a start state of the 3D vector level. Thus generation links (shown as shaded lines is Figure 2) depict hypothesis creation while verification links (shown as dark lines) depict hypothesis verification.

Generation KS links are the main difference between recognition graphs and decision trees. Decision trees are devices for verifying hypotheses. Recognition graphs describe hypothesis generation as well as hypothesis verification. The start states at any level of abstraction correspond to the generation KSs that can create hypothesis of that type (Unlike decision trees, each level of a recognition graph may have multiple start states.). The verification KS links describe how those hypotheses can be verified. Hypotheses that prove sufficiently reliable can then be used as arguments to generation KSs, which create hypotheses at the next level of abstraction.

Recognition graphs can represent a variety of strategies. Bottom-up strategies are represented by having the generation KS links point "up" the hierarchy; top-down strategies are implemented by having them point the other way. Most strategies are mixed, in that they have generation links going both up and down the hierarchy. For example, line segment hypotheses can either be extracted from the image (bottom-up) or predicted by a road hypothesis (top-down). Refinement strategies can be represented through seemingly circular links representing iterative processing, as when the perspective KS a projects a 3D pose hypothesis, creating 2D line segment hypotheses which are compared to lines extracted from the image and used to generate a more accurate pose hypothesis (which, if verified, can be used to revise the 2D line segment hypotheses...).

Once completed, recognition graphs are used to identify objects in the intuitive way. Hypotheses begin at starting states determined by the generation KSs that created them.) The options at that knowledge state represent the different verification KSs that can be applied to the hypothesis. If the recognition graph has been optimized by SLS, then the options will be ordered in terms of their *gain* (see Section 2.2.2 below). The verification KS with the highest gain at the current knowledge state will be applied to the hypothesis and will return an endorsement about it. This endorsement effectively moves the hypothesis to a new knowledge state, where the process is repeated.

Anytime a hypothesis reaches a knowledge state with a generation KS link, the generation KS is applied to the hypothesis. This creates zero or more new hypotheses (usually at a different level of abstraction) which are placed at the appropriate starting state. These hypotheses are then pursued concurrently with the original hypothesis. The interpretation is finished when all hypotheses have reached a state from which no action can be taken. At that point all 3D pose hypotheses at positive goal states (see Section 2.2.2) are presented to the user as object instances.

Finally, recognition graphs are not actually graphs but hyper-graphs. Many knowledge sources calculate relations between hypotheses. Consider, for example, a knowledge source that tests if a 2D line segment lies on the boundary of a region. This KS operates on two hypotheses at once, and potentially returns supporting evidence about both of them. The two hypotheses start from different knowledge states, and in this case from different levels of abstraction. If the relationship holds between them, both hypotheses will be moved to new knowledge states. In this case the line hypothesis would be moved to a state indicating that it is supported by region hypothesis, and the region hypothesis would be moved to one reflecting the support of the line. Thus the KS that tests for this relation is a hyper-link in the recognition graph between two pairs of knowledge states.

2.2 The Knowledge Compiler

SLS begins the learning process by building a recognition graph that includes every possible option at each knowledge state. It then uses this graph to interpret the training images, exhaustively applying every verificaiton KS to every hypothesis it can possibly generate. By monitoring this process, SLS is able to estimate the expected cost of every KS. In addition it learns the probability of every possible outcome of each verification KS, and it notes which hypotheses were created by which generation KSs. This information will be used to 1) prune unnecessary generation KS links from the graph, so that only well-supported hypotheses are used to create new hypotheses, and 2) order the verification KSs at each knowledge state according to their gain, so that run-time monitor knows which KS to apply (if any) to minimize the expected cost of recognition.

2.2.1 Hypothesis Generation

The next step is to learn how to generate hypotheses. The user's object description may include generation KSs that are not necessary to recognize the object. Moreover, most of the necessary generation KSs should only by applied to well-supported hypotheses. SLS therefore seeks to constrain the application of generation KSs so that as few hypotheses as possible are generated from the training images, but all the correct hypotheses are still created.

SLS determine which hypotheses generated from the training images are correct by comparing them to the interpretations supplied by the user. The system then takes each correct hypothesis and traces back the generation KSs it depends on. For example, if a correct 3D pose hypothesis was generated by matching the model vertices to a particular set of 2D points, then the pose hypothesis is dependent on the matching KS. It is also dependent on whatever generation KS created the 2D points, and any generation KSs needed to create its arguments, etc. In this way, the generation KSs needed to create each correct hypothesis are determined.

The result of tracing back a hypothesis' dependencies is an AND/OR tree, since multiple KSs may generate the same hypothesis. For example, the 3D pose hypotheses imagined above might also have been generated by an absolute orientation KS applied to a set of 3D points. If so, the 3D pose hypothesis is dependent on either the matching KS and those KSs needed to extract the 2D points or the absolute orientation KS, along with whatever KSs are needed to generate the 3D points.

Since the system's goal is to generate all the correct hypotheses, it collects the dependency trees of all the correct hypotheses into a single tree by ANDing their top nodes together. By definition, any set of generation KSs that satisfies this tree will generate all the correct hypotheses. The system's job is to find the particular set of KSs that satisfies this tree while creating as few incorrect hypotheses as possible. SLS finds this set by converting the AND/OR tree into disjunctive normal form (DNF). Each subterm in the DNF expression represents a set of generation KSs that satisfies the tree. Each of these subterms is tested, and the one that generates the fewest incorrect or un-

necessary hypotheses is selected. Every generation KS not in this subterm is discarded.

In addition to determining which generation KSs should be applied, the system must also determine when they should be applied. Returning to our example above, if every 2D point needed by the matching KS to create a correct 3D object hypothesis received a "high-contrast" endorsement from a verification KS, then the matching KS should only be applied to point hypotheses with this endorsement. By tracing back the dependencies of the correct hypotheses, the system can determine what endorsements are shared by the hypotheses needed for each generation KS. The generation KS is then constrained so that it is only applied from knowledge states which include these endorsements. All generation KS links which begin at states which do not have the required endorsements are pruned from the recognition graph.

2.2.2 Hypothesis Verification

Once SLS has learned how to generate hypotheses, its next task is to learn how to verify them. This in turn can be divided into two subtasks, classification and optimization. In the first, SLS learns how to verify hypotheses. In the second it learns how to verify them efficiently.

SLS learns to classify 3D pose hypotheses by building an ID3-style decision tree classifier ([2]). The classifier's task is binary; either a hypothesis has enough support to satisfy the user's accuracy requirements or it doesn't. The training instances are the hypotheses generated from the training images; the features are the presence (or absense) of particular endorsements on those hypotheses.

Since the classifier is trained on sets of endorsements, it can be used to classify knowledge states. Any knowledge state at the 3D pose level that is accepted by the classifier is a positive goal state. All hypotheses that reach positive goal states at the 3D pose level are included in the final interperetation. Knowledge states that are not positive goal states and which do not lie on any path to a positive goal state are negative goal states. A hypothesis that reaches a negative goal state is immediately discarded by the system. States which are neither positive nor negative goal states are undetermined, and indicate that the hypothesis needs to be verified further.

Positive and negative goal states are also defined at lower levels of abstraction. Hypotheses such as regions, 2D line segments and 3D line segments are never included in the final interpretation. Their only purpose is the role they play in creating 3D pose hypotheses. Consequently an intermediate-level hypothesis is verified when it has enough supporting endorsements for it to serve as an argument to a generator KS (as determined in the previous section). A positive goal state at an intermediate level of abstraction is therefore any state that satisfies the prerequisites of a generation KS[1]. A negative goal state is one which does not lie on any path to a positive goal state. As before, knowledge states at the intermediate levels that are neither positive nor negative goal states indicate that more evidence needs to be acquired.

Once the goal states have been determined at each level of abstraction, SLS selects and orders the verification KSs so as to minimize the expected cost of verification. Since SLS has already minimized the number of hypotheses it generates, the result is a strategy that approximately minimizes the total expected cost of recognition.

SLS chooses verification KSs by computing the expected cost of reaching a goal state (either positive or negative) from each knowledge state in the recognition graph. To do this, it needs to compute the expected cost of reaching a goal state from a knowledge state using each of the available options. For example, if two verification KSs (*VKS1* and *VKS2*) can be executed from knowledge state n, SLS will calculate the expected cost of reaching a goal state if *VKS1* is applied first, and the cost of reaching a goal state starting with *VKS2*. The cost of reaching a goal state from state n would then be the cost of reaching a goal state using the cheaper of the two options.

More formally, we refer to the cost of promoting a hypothesis from knowledge state n to a goal state as the Expected Decision Cost (EDC) of state n. We denote the expected cost of reaching a goal state from state n using verification KS k as $EDC(n,k)$. Since verification KSs return endorsements, we refer to the possible outcomes of a verification KS k as $R(k)$, and the probability of an endorsement e being returned as $P(e|k,n), e \in R(k)$.

SLS calculates the EDC's of knowledge states starting with the goal states and working backwards through the recognition graph. Clearly, the EDC of a goal state is zero:

$$EDC(n) = 0, \quad n \in \{goal\ states\}. \tag{1}$$

The expected cost of promoting a hypothesis to a goal state using a particular KS is:

$$EDC(n,k) = C(k) + \sum_{e \in R(k)} (P(e|n,k) \times EDC(n \cup e)) \tag{2}$$

where n is the knowledge state expressed as a set of endorsements, and $C(k)$ is the estimated cost of applying k (if k can return the empty set of endorsements, this equation needs to be extended slightly).

The EDC of a knowledge state, then, is the smallest EDC of the knowledge sources that can be executed at that state:

$$EDC(n) = \min_{k \in KS(n)} (EDC(n,k)) \tag{3}$$

where $KS(n)$ is the set of KSs that can applied at node n.

Equations 2 and 3 establish a mutually recursive definition of a node's expected decision cost. The EDC of a knowledge state is the EDC of the optimal KS application at the state; the EDC of a KS application is the expected cost after applying the KS, plus the cost of the KS. The recursion bottoms out at goal nodes, whose EDC is zero. Since every path through the object recognition graph ends at a goal node (either positive or negative), the recursion is well defined.

If all verification KSs operated on a single hypothesis, then the cost of verification could be minimized by always selecting the verification KS that minimized a knowledge state's EDC. As mentioned before, however, relational KSs may provide support for more then one hypothesis at a time. It is therefore necessary to define the *gain* of a KS as the sum of the expected drop in EDC for each of its arguments, minus the cost of the KS. For a

[1] To be precise, any state such that 1) it satisfies the prerequisites of a generation KS and 2) no endorsements can be added to it to make it satisfy the prerequisites of another generation KS.

Figure 3: A typical New England road scene containing a cautionary road sign.

single-argument verification KS, the gain is defined as:

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

For a verification KS that relates two hypotheses to each other, the gain is defined as:

$$Gain(n1,n2,k) = EDC(n1) - \sum_{e1 \in R(k)} (P(e1|n1,k) \times EDC(n1 \cup e1)) + EDC(n2) - \sum_{e2 \in R(k)} (P(e2|n1,k) \times EDC(n2 \cup e1)) - C(k)$$

The optimal overall strategy for verifying hypotheses is the one that minimizes the expected cost. The expected cost of verification is minimized by selecting, at each knowledge state, the KS with the maximum gain.

3 Recognizing Road Signs in SUA

We will illustrate SLS' approach with a cursory example of learning to recognize road signs from the image shown in Figure 3. Although strategies learned from a single sample are necessarily over-specific, the example should demonstrate how the system works.

SLS begins by making a recognition graph out of the user's object description. In the case of a road sign, the object description might include generation KSs for vanishing point analysis, scale estimation from distance and orientation, 2D→3D point matching, region segmentation, 2D point extraction and straight line extraction (the last two operating on a "window of interest" defined by a region). It might also include verification KSs to test if a region is 1) yellow or 2) four-sided. The vertices of a wire-frame model of a road sign are included as a parameter of the 2D→3D point matching KS.

During training, 3D pose hypotheses were created three different ways. All three segment the image and classify the regions to see which match the expected color of a road sign. The first method extracts lines from the image near the region hypothesis, performs vanishing point analysis to estimate its 3D orientation, and then judges its distance from its size and orientation, producing a 3D pose. The second method extracts 2D points from the image under the region hypothesis and finds the pose directly by 2D→3D point matching. The third method is like the second, except that it uses the corners of the region hypothesis rather then extracting points directly from the image.

Unfortunately, the pose created by the first method was incorrect (vanishing point analysis is unreliable when the 2D lines are nearly parallel, as in Figure 3). Methods two and three, however, both discovered the sign's correct pose. The dependency tree for this hypothesis was therefore an OR of methods two and three. When given a choice between alternate generation techniques, SLS chooses the one that generates the fewest total hypotheses. In this case it picked method three, since the 2D point extraction found five 2D point hypotheses (one of which corresponded to the tip of the arrow), while the boundary approximation routine produced only four. SLS therefore kept only the region segmentation and point matching KSs. All generation KS links corresponding to other generators were pruned from the recognition graph. SLS also noticed that the point matching KS should only be applied to the corners of regions with the endorsement "good-color-match". All point matching KS links not starting from a knowledge state with this endorsement were therefore removed from the graph.

Next SLS learns to verify hypotheses. Since every hypothesis produced by method three was correct, the ID3-style classifier notes that the start state at the 3D pose level corresponding to the 2D→3D point matcher is a positive goal state. At the start state of the region hypothesis level, it notices that the KS that tests if a region is four-sided has negative gain, while the color test has a gain of zero. It therefore decides that the color KS should be executed from that state. Since every result from the color KS leads directly to a positive or negative goal state, the four-sided KS is never executed.

4 Conclusions

An object's description determines the most efficient and accurate method for recognizing the object. An important problem in vision, therefore, is learning how to recognize an object from its description and a set of examples. An effective object recognition system must learn which KSs to apply, when to apply them, and how to integrate their results.

We solve the problem of selecting which knowledge source to execute when by selecting the KS that minimizes the expected cost of recognition, subject to accuracy constraints imposed by the user. We select these KSs at compile-time in a two-step process. The first step minimizes the number of incorrect or unnecessary hypotheses generated. The second step learns how to verify the remaining hypotheses by building a classifier and then ordering the verification KSs so as to minimize the cost of verification. The resulting recognition strategy in embedded in a recognition graph.

References

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

[2] J.R. Quinlan. "Induction of Decision Trees", *Machine Learning,* 1:81-106 (1986).