A COMPUTER SYSTEM FOR VISUAL RECOGNITION USING ACTIVE KNOWLEDGE*

Eugene C. Freuder+
Computer Science Department
Indiana University
Blooraington, Indiana 47401

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643. The report is based upon the author's Ph.D. dissertation for the Department of Electrical Engineering and Computer Science, M.I.T., 1975.

'Current address: Department of Mathematics and Computer Science, University of New Hampshire, Durham, N.H. 03824.

OVERVIEW

THIS WORK IS CONCERNED WITH VISUAL RECOGNITION, AND WITH LARGER ISSUES OF DESCRIPTION AND CONTROL

The immediate objective of this work Is a computer system that will recognize objects in a visual scene, specifically hammers. The computer receives an array of light intensities from a device like a tv camera. It is to locate and identify the hammer if one is present. The system, SEER, is directed at the visual variety of unprepared everyday objects, not stylized models.

The computer must produce from the numerical "sensory data" a symbolic description that constitutes its perception of the scene. Of primary concern is the control of the recognition process: what to do, how and when to do it. These decisions should be guided by the partial results obtained on the scene. If a hammer handle is observed this should suggest that the handle is part of a hammer and advise where to look for the hammer head. The particular knowledge that a handle has been found combines with general knowledge about hammers to influence the recognition process. This use of knowledge to direct control is denoted here by the term active knowledge.

A descriptive formalism is presented for visual knowledge which identifies the relationships relevant to the active use of the knowledge. A control 'structure is provided which can apply knowledge organized in this fashion actively to the processing of a given scene.

VISUAL KNOWLEDGE IS ORGANIZED TO ENCOURAGE ITS ACTIVE USE

Given a commitment to active knowledge how do we implement it? When SEER discovers that "region R is bar shaped" how does it use this information? General visual knowledge is organized to respond to these questions.

A data base object exists representing "bar shaped". This object knows how a bar shape is useful for acquiring further knowledge of a region. It knows, for example, that hammer handles are bar shaped. This knowledge is represented by a link between data base objects representing bar shaped and hammer handle:

HANDLE——BAR-SHAPED

ESTABLISHED RESULTS ARE EXPLOITED TO MAKE FURTHER SUGGESTIONS

When a particular region R is found to be bar shaped, this knowledge is exploited as follows. The bar shaped element of general knowledge is consulted. The link to hammer handle provides the suggestion that R is also a hammer handle. This suggestion becomes an object in the particular knowledge data base. It begins as a "conjecture", but eventually may become a "success" or a "failure". (The fact that R is bar shaped is a successful particular knowledge object.) A link is placed between BAR-SHAPED R and HANDLE R corresponding to the general knowledge link between BAR-SHAPED and HAMMER HANDLE:

HANDLE R——BAR-SHAPED R

BAR-SHAPED is also linked to HEAD in the general knowledge: the exploitation of BAR-SHAPED R will also suggest HEAD R, and the appropriate particular knowledge conjecture and link will be created:

HANDLE R\
BAR-SHAPED R
HEAD R /

GENERAL VISUAL KNOWLEDGE AND PARTICULAR SCENE RESULTS RESTDE IN INTERACTING NETWORKS

We have then two knowledge structures, which we will abbreviate as the GK (general knowledge) and the PK Cparticular knowledge) structures. Both are networks: the nodes represent items of visual knowledge, objects, properties, relationships, and the links indicate how these items help establish each other.

The knowledge involved is not very esoteric. It is based on functional definitions: a hammer requires a handle to hold and a head with which to hit; the two should be set at right angles to transfer a swinging motion into a blow; the head requires a striking face at one end; it should be flat to contact the nail easily; and so on. The concern is with the organization of this knowledge. Standard descriptive networks, where nodes represent parts and links represent properties and relationships, are good for passive matching. SEER's networks facilitate active knowledge.

The GK represents potential programs for recognizing specific instances of the GK concepts. We write these programs by representing possible descriptions of the concepts in a network form. These descriptive nets seem a natural formalism for the procedural embedding of knowledge [7] In the domain of visual recognition (as opposed to theorems, for example). Similarly the PK represents, at any point of processing, the current state of description of the scene, and the state of the specific processes chosen to work on it.

The interactions within and between these structures implement active knowledge, and we have seen a basic example: a PK success consulting GK to suggest further PK conjectures.

CONJECTURES ARE ESTABLISHED BY EXPLORATION

We have seen the basic mechanism for exploiting results, but how do these results get established; once a suggestion is made, how is it pursued? The same link that indicates that a bar shaped region may be a handle, also says that if

we wish to establish a region as a handle, we should see if it is bar shaped. Thus, analogous to exploitation there is an exploration process. When the PK conjecture HANDLE R is explored, it consults the GK object HANDLE, which follows the link to BAR-SHAPED, providing the suggestion BAR-SHAPED R, which enters the PK as a conjecture linked to the HANDLE R conjecture:

HANDLE R——BAR-SHAPED R

The links have a direction: if B helps to establish A we say that B is "below" A. Exploitation involves looking upward along links, exploration downward.

Exploration of HANDLE R also suggests LONG-AND-THIN R:

HANDLE R LONG-AND-THIN R
\BAR-SHAPED R

Let us say that the two properties, bar shaped and long-and-thin, are sufficient to define a hammer handle; the GK knows this. Now if either of the new conjectures, BAR-SHAPED R or LONG-AND-THIN R, are established, they will be exploited. We did observe earlier that exploitation of BAR-SHAPED R involves creation of a new PK object for HANDLE R. However, obviously this will not be necessary now, as HANDLE R already exists as a PK conjecture. What will happen rather is that the HANDLE R conjecture will be found and told that one of the properties it needs to succeed has been established. The GK has told HANDLE R to wait for two results (it is an "AND gate" if you like); it will now count one and wait for the other. When that comes, HANDLE R will in turn succeed and be exploited.

That is how HANDLE R gets established; but what about BAR-SHAPED R and LONG-AND-THIN R, which must be established first? Clearly we can repeat the exploration process, but also the process must terminate somewhere. Exploration of LONG-AND-THIN R leads to three further conjectures:

LENGTH R
LONG-AND-THIN R -WIDTH R
LONG-AND-THIN-COMP R

HAMMER R

BAR-SHAPED R

Let us suppose, as might be the case, that LENGTH R and WIDTH R are already present and successful in the PK. The exploration of LONG-AND-THIN R would not duplicate them; they would be found and linked to, and the new links exploited by notifying LONG-AND-THIN R that two of its requirements have been met. (The actual numerical values of the length and width of R will have been stored as properties of the corresponding PK elements.)

LONG-AND-THIN-COMPUTATION R, when explored in turn, does not pursue further conjectures, but actually compares the length and width with a threshhold and either succeeds or fails on the spot. A success will complete the establishment of LONG-AND-THIN R. All PK trees eventually lead down to such terminal computation nodes.

GLOBAL MONITOR AND PRIORITY MODULES CHOOSE WHICH SUGGESTIONS TO EXPLORE

We see by now that there will be a lot of suggestions floating around, proposed by results or pursued by hypotheses. These suggestions form a "pool" of processing possibilities; this pool concept is important because it eliminates the need for suggestions to be taken up or rejected immediately. Instead they can accumulate and "compete", as further results improve our ability to discern the most promising lines of inquiry. We know how to explore and then exploit the suggestions, but the question remains: when to explore them, or indeed, which to explore.

Lurking in the background are critical doubts. Can we choose intelligently, or will the number of suggestions "blow up" to an unmanageable number? The suggestion making processes, exploration and exploitation, are basically local operations, centered around the individual results and pieces of general knowledge. One can argue the advantages of such a uniform, modular structure; but how is chaos avoided: Is some global regulation imposed?

Basically, we are hoping for something of a "Waltz convergence effect" [21] on the pool of suggestions. (However, it should be noted that this research does not focus on the problems raised by very large data bases. These will probably require further means of partitioning our knowledge, e.g. into frames [12]; SEER's basic interest is fruitful interaction within our knowledge.) Waltz' program may be viewed as carrying a large number of line labelling conjectures, based on partial evidence, forward in parallel. Since the legitimate combinations of labels, among parts of the scene, were limited, as processing proceeded eventually more possibilities were ruled out than were added, and only the one or two complete labelllngs for the scene were left.

In SEER a number of conjectures are also carried forward in parallel. Here too new results can contradict old conjectures. However, we emphasize more the positive effect that new results can have in encouraging previous conjectures, directing our attention to the most promising and the most efficient suggestions. A priority system is employed for this purpose. A monitor uses the priority information to determine the order in which suggestions will be explored. These are the global elements of the system, which evaluate the local activity.

The priority of a PK conjecture is basically a cost/benefit analysis. The likelihood of a conjecture holding true is compared with the expected difficulty in establishing it. These figures can change as results accrue. The conjecture that R Is a hammer is initially difficult and not too likely (though its high "interest" for us can also be taken into account). When R is found to have a hammer handle, the difficulty of the hammer conjecture goes down while its likelihood goes up, thus its overall priority increases. (It is, again, the exploitation of the hammer handle success which induces that change in priority.)

To the monitor the PK looks something like a mountain range. The peaks represent those nodes which have as yet nothing above them in the link structure. Below each peak are linked nodes, and nodes linked to them in turn, continuing until we reach as yet unexplored nodes, with no further

links below them. This structure of nodes below the peak represents the state of the investigation of the peak node. These structures blend into one another, especially in the "foothills", e.g. CONVEX R is likely to appear in several investigations. The monitor must first decide which investigation is most promising, then which suggestion within that investigation to explore next. It chooses to further investigate the peak node of highest priority. Choosing the next suggestion to explore is not quite so simple, but basically we want to explore the easiest, least likely suggestion first, in order to resolve the investigation most quickly.

PROCESSING CYCLES THROUGH SUCCESSIVE PK STATES

After the initial passive stage, in which regions of potential interest are found, an initial PK state is established with a few "seed suggestions" about prominent regions. These are "primitive" conjectures like CONVEX R. CONVEX is primitive in the sense that there Is essentially nothing below it in the GK network except properties like area and boundary which always can be established. From this point on, processing entails active knowledge, use of previous results and general knowledge.

Processing consists of a sequence of cycles, which carry us from one state of the PK to the next. Each cycle begins with the exploration of a suggestion chosen by the monitor. There may follow one or more exploitations if successful results are obtained. (A similar process makes use of failures.) Exploration and exploitation will change the PK: new conjectures may be added, new links, priorities may change, conjectures may become successes or failures. At the conclusion of the cycle, control returns to the monitor to begin the next cycle.

Thus we have continual review of local interactions by the global mechanisms. SEER cannot continue blindly down one avenue, but must evaluate the impact of results in a wider context. Also while activity within a cycle is local in the sense that it is centered on individual nodes, this activity is not at all parochial. There is not, for example, the usual "master/slave" relationship between processes. A result does not report only to the calling process, but is a gregarious fellow who happily tells anyone who might be interested, to whom he might be useful, of his good fortune. Even a success in one investigation, may when exploited have a greater effect on another investigation; the monitor can observe this and switch its attention to the second investigation. In any case, as investigations succeed, they in turn spawn larger investigations, until eventually we reach a satisfactory level of recognition, e.g. find the hammer.

METHODS ADVISE HOW TO ESTABLISH RESULTS

We now have an idea of SEER's approach to what to do and when to do it. There remains the important question of how to do it. SEER makes suggestions about what to do, it also gives advice on how to do it. Actually the primary Implementation of advice is as a form of suggestion. SEER suggests methods for establishing results.

A method is also a node in the GK or PK structure. The discussion up until now may have given the impression that these structures are intertwined trees of AND gates. Actually they involve AND/OR trees. An AND node links to a set of subnodes which together define or establish the node, a set of properties or parts; we have seen examples. An OR node links down to a set of subnodes each of which Is an alternative method for establishing the node. Thus HANDLE may actually link to HANDLE-METHOD-1, HANDLE-METHOD-2, etc. HANDLE-METHOD-1, for example, may involve finding the handle when we already have the head to advise us where to look. When the head is found the exploitation of that result will suggest HANDLE-METHOD-1. This approach to advice is very explicit; advice on how to do something is a suggestion of a specific method for doing it.

EXAMPLE

AN EXAMPLE WILL ILLUSTRATE MORE OF SEER IN OPERATION

As in the AND/OR business, this discussion was begun by simplifying SEER's structure and operation as much as possible, and the picture has only been complicated as absolutely necessary. For example, we eventually observed that exploitation involved updating priorities as well as creating new conjectures; however, we avoided the fact that priority activities are carried out by following a set of priority links which are distinct from the links that indicate how items define one another. (Most often, these two link types will occur together; however, they do represent distinct relationships which have been extracted from the basic concept of "relevance" between pieces of visual knowledge.) This is not the place for full detail; however, an example of SEER's operation is in order which will provide a feeling for the types of problems and solutions which arise.

AN AFFINITY PASS PROVIDES AN INITIAL SET OF REGIONS

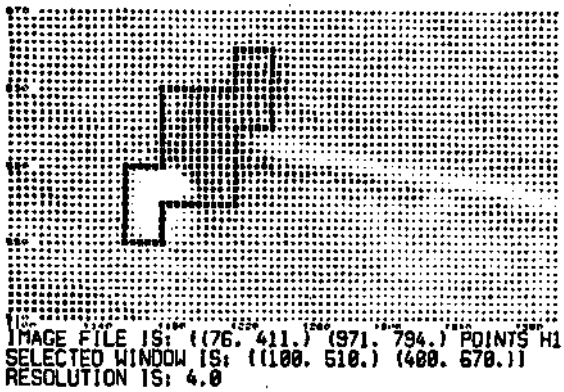
Consider a scene consisting of a ball peen hammer on a wooden work bench. First SEER makes a region growing pass over the scene using what we term an affinity process [5]. Briefly, rather than seeking absolute uniformity, which may not be present in the surfaces of a realistic object, the process attempts to find regions which are relatively homogeneous, whose pieces seem to hang together more than they seem to belong with other neighboring pieces of the scene. Rather than finding one division of the scene, the process builds up a tree of potentially interesting regions. Below each region in the tree are the subregions which were merged to form it. The merging process works upwards from a initial division of the scene into small cells, forming successively larger regions.

SEER CHOOSES A PROCESSING SEQUENCE, BASED ON THE GK STRUCTURE AND DEVELOPING RESULTS IN THE SCENE: SEER "ZIGZAGS" GRADUALLY UPWARD FROM THE SEED SUGGESTION THAT REGION 196 IS CONVEX, TO A CONJECTURE THAT THE REGION IS A HAMMER HEAD

The regions that "stand out" in the scene will be found near the top of the tree, and so we begin by making primitive conjectures about these. In this case the initial PK will consist of seed

conjectures for regions R196, R191, R185, R140. SEER explores the conjecture CONVEX R196, region 196 is convex. Fig. 1 shows this region superimposed on the scene.

FIG. I. Region 196.



The exploration of CONVEX R196 leads to the PK structure:

```
                / CONVEX-NEEDS R196
CONVEX R196
                \ CONVEX-COMPUTATION R196
```

CONVEX-NEEDS represents the Items needed to perform the computation CONVEX-COMPUTATION which will judge convexity. The computation cannot be made until these items are found, so the GK node CONVEX, and this PK instance, CONVEX R196, are "serial" nodes. This means that the process of choosing which suggestion to explore next is constrained at this point. If SEER wishes to further investigate CONVEX R196, it must establish CONVEX-NEEDS R196 before it can explore CONVEX-COMPUTATION R196. CONVEX-NEEDS is explored in the next cycle and the PK investigation of CONVEX R196 expands:

```
                                  /CONVEX-HULL R196
              /CONVEX-NEEDS R196
             /                    \AREA R196
CONVEX R196
             \
              \CONVEX-COMPUTATION R196
```

Both AREA and CONVEX-HULL must be found in order to satisfy the AND node CONVEX-NEEDS RI96; however, either may be found first in this case. In writing up this portion of the GK, we do not need to specify a serial order, but can program in parallel, in some sense. Neither do we need to consider explicitly all the different possible execution time conditions that might influence the decision on which to explore first: "if A has been found, do B, however if C..." The relevant conditions will be reflected in the state of the PK, and the ad hoc local decisions we might make have been generalized and given to SEER's priority and monitor modules. These can take a global view, based on the state of processing at decision time.

If AREA R196 has been established already, attention will automatically focus on CONVEX-HULL R196. Perhaps neither is yet a success, but CONVEX-HULL R196 has been partially established, a subnode has succeeded. The exploitation of that subnode will have improved the priority of CONVEX-HULL R196. This may influence the monitor to choose to explore CONVEX-HULL R196 before AREA R196. Of course, in this example no results have yet been obtained, and SEER chooses which node to explore based on initial priorities found in the GK. After SEER begins developing one branch, however, it is not locked into that direction. Perhaps an easy method for obtaining the convex hull fails, priorities shift, and the monitor can shift attention to finding the area. As it happens, finding the area and convex hull are rather straightforward, and R196 is in fact determined to be convex.

SEER now begins a typical "zigzag" upward flow. Exploiting CONVEX R196 leads to several suggestions being placed above CONVEX R196 in the PK. One of these is chosen and investigated. Several cycles of explorations expand the investigation back downward in the PK. Finally the investigation succeeds and exploitation again expands the PK upward. Once again an investigation is chosen to explore, and so it goes. This basic flow is, of course, subject to many alterations; results within an investigation can also propose new investigations; most significantly SEER can choose to shift attention between investigations, just as it can between branches within an investigation.

Notice that the processing flow is neither strictly "top down" nor "bottom up" but more "middle out" [8]. A top down control structure can direct its computations very efficiently. In the extreme, we know precisely what we are looking for and can employ a sensitive template to merely verify its presence. A top down system, however, lacks generality, because it begs the question of how to choose the starting point in an environment with multiple possibilities. SEER attempts a compromise. A result proposes a higher hypothesis, which can then be used as a context for proceeding downward. Our view of knowledge elements as "little men" [17] who know how to explore and exploit themselves, facilitates this middle out model.

SEER soon establishes that region 196 is a "bar", a basic shape category. This result proposes several interesting suggestions: R196 is a hammer handle, an occluded hammer handle, a hammer head. The first two are explored but quickly lead to failures.

INSTANTIATION FUNCTIONS ARE USED TO GENERATE ARGUMENTS FOR SUGGESTIONS: A REGION IS FOUND FOR THE FACE OF THE HAMMER HEAD, AND THEN ANOTHER IS CONJECTURED FOR THE HAMMER ITSELF

The GK indicates that to establish that R196 is a hammer head, SEER must establish that R196 has a striking face. Here we are forced to confront another major factor which was glossed over earlier: the generation of arguments. Up until now we have used the same argument, R196, in all our PK statements. There has been no problem in making new suggestions; exploring CONVEX-NEEDS R196, SEER looks at the GK node for CONVEX-N, finds AREA node below it, and suggests AREA R196, what else? But IS-A-HEAD R196 cannot suggest TS-A-HEAD-FACE R196.

Actually the GK nodes also have arguments, variable arguments: CONVEX-NEEDS X, AREA Y, IS-A-HEAD Z. Attached to the links in the GK are functions, instantiation functions, which indicate the relationship between the arguments of connected nodes. Very often the instantiation function is simply the identity function, I. Thus when SEER explores CONVEX-NEEDS R196, it consults CONVEX-NEEDS X in the GK, follows a downward link to AREA Y, and then applies the instantiation function I to the value associated with X in the PK, R196, to obtain a value for Y:I(R196) = R196. The resulting statement AREA R196, is one of the suggestions to be made in exploring CONVEX-NEEDS R196.

These instantion functions can be arbitrarily complex; however, we try to keep most of the work involved on the system in the nodes. Thus a common GK structure is generate and test:

```
          / FOO-GENERATE A
F00   A
          \ FOO-TEST A
                        \ ZOT B
```

Here the FOO-GENERATE A node computes B and stores it as a property of the node. The instantiation function on the GK link from FOO-TEST X to ZOT Y picks up B from the property list. B is then tested for the ZOT property.

Often an additional complication is the need to generate several alternatives for testing. If we search for the hammer head face in the original affinity region tree, for example, we want to allow for several tries at finding a region with the right properties. There are various options. The function on the link between FOO-TEST and ZOT can be multivalued, for example. Though ZOT Y only appears once in the GK, several instances of it may be suggested when we explore a FOO-TEST node in the PK. (FOO-TEST will be an OR node obviously.)

It is often painful to generate every option at once, however, and a looping mechanism must be employed. There are methods of implementing these in SEER, and one is in fact utilized in searching for the hammer head face in this case. However, they are a bit awkward at present, and such technical details should not delay us here. Suffice it to say that the face of the hammer is indeed found in the affinity tree, region 145. Actually this tree search is just one method for finding a hammer head face: HAS-A-HEAD-FACE-METHOD-1 R196 was the successful node. In general other, techniques are possible for generating desired regions. We are about to see one of these.

When IS-A-HEAD R196 succeeds, it suggests the presence of an encompassing hammer, and advises a method for finding the hammer handle.

Actually, there is an intermediary step. IS-A-HEAD R196 suggests HAS-A-HEAD DR1, region DR1 has a hammer head as a subregion. This again is a situation where a new argument is required. One of the obvious things to do is to generate a 'dummy" argument. DR1, "dummy region one", stands for the hammer region which SEER hopes to define by the union of R196 and another region for the handle, yet to be found. The instantiation function in the GK link between IS-A-HEAD and HAS-A-HEAD supplies the argument DR1.

RELATIONSHIPS PROVIDE ADVICE: THE ESTABLISHED HEAD HELPS FIND THE HANDLE

HAS-A-HEAD DR1 succeeds immediately; normally a method for establishing HAS-A-HEAD will need to prove proper relationships hold between the head and the hammer region (or the handle region). However, this is obviously not an issue now. When the handle is found the proper relationships will be checked. Better yet we will use these relationships to guide the search for the handle. Thus the exploitiation of HAS-A-HEAD suggests HAS-A-HANDLE-METHOD-2, a method which uses the head to find the handle. Also suggested is IS-A-HAMMER-METHOD-1 DR1.

Investigating 1S-A-HAMMER-METH0D-1, SEER suggests several methods for finding the handle but method two obviously has priority at this point:

```
HAS-A-HEAD DR1

IS-A-HAMMER-METH0D-1 DR1

                                  /HAS-A-HANDLE-METHOD-1 DR1
HAS-A-HANDLE DR1                  -HAS-A-HANDLE-METHOD-2 DR1
                                  \HAS-A-HANDLE-METH0D-3 DR1
```

This method is based on a program developed by Tomas Lozano-Perez which uses an ATN grammar to look for regions with specified intensity profiles [101. The program requires a starting point and a direction in which to look, along with a specification of the type of intensity profile being sought. SEER uses the head, R196, which it has already found, and the expected relationship between hammer heads and hammer handles to determine a starting point and direction to send to Lozano's program, along with the characteristic curved profile of a plot of light intensities across the width of a hammer handle. Having hypothesized what we are looking for, and guided in where to look, we are able to use this specialized technique, and it is successful. A region NR1, "new region one", not in the original affinity tree, is returned. A few additional checks ensure that it will serve for the hammer handle.

Notice that the basic relationships between hammer and head do not need to be checked, they have already been verified in the process of generating NR1. Contrast this with the distinct passes of some passive knowledge systems, a descriptive phase would find regions and determine relationships, then a recognition phase would match these relationships with those in a model. In SEER description and recognition tend to merge together. The importance of this merging is one reason SEER was designed to work directly from camera input, as opposed to a "hand coded" symbolic description of the scene. We do not wish to beg the key question of acquiring a perceptual description from the sensory data; rather we wish to bring recognition level knowledge to bear on it. The interaction with raw data keeps us honest and ensures that our results have a bearing on real visual problems.

SEER now has the entire hammer and the recognition is complete.

EVALUATION

SEER PERMITS A FLEXIBLE RESPONSE TO THE VARIETY IN REALISTIC SCENES

It should be emphasized that the above example does not present the way that SEER recognizes hammers. One of SEER's aims has been to deal with realistic scenes and to provide a system with the flexibility to deal with the variety that realistic scenes hold. (Even a single hammer can be given many different appearances by varying its orientation, background, lighting.) SEER's modular organization allows us to program basic alternatives, then add others incrementally as we expand SEER's competence. SEER's mode of processing, instantiating successive pieces of the modular GK structure, which is both program and description, allows it to enter the "hammer program" at a variety of points and proceed in a variety of paths through the structure, exploring and establishing different subsets of the "hammer description" in different orders—all in response to the results being obtained on a given scene.

Thus, in the example we discussed, SEER began by working up to a head recognition, then moved back down to find the handle. The handle was narrow and angled sharply, its wooden texture blended into the workbench background. It might have been hard to recognize; it was not found in the initial affinity pass. (A "finer" pass with this process might have succeeded, but could be prohibitively expensive in time and space.) However, with the advice provided by the previously found head, the handle was found, and relatively easily.

We could have discussed another scene in which the light wooden handle stands out while the rusty head blends into the dark background. In this scene SEER identifies the handle first, using a different method than the one employed here, and then goes on to find the head.

SEER ANALYZES AND IMPLEMENTS ACTIVE KNOWLEDGE

SEER is a programming language or system that provides a framework for active knowledge programming, in the sense that a language like PLANNER [7] identifies and implements various processing methods suitable for problem solving activity. Some previous vision systems pioneered by providing examples of active knowledge types of activity, but were not extensible or general in their approach. The Wizard line finding system [23], for example, used lines, as they were found, to suggest the location of further lines. This was accomplished basically by assuming a parallelipiped environment and trying to find lines that completed parallelograms. However Wizard could not supply ready or direct answers to questions like: what about wedges; how do I add a facility for dealing with wedges to the system and integrate it with what is already there? What is the general nature of the suggestion process; how can I apply it to another body of knowledge; what kind of analysis of the knowledge is required; in what ways can I profit; what are the basic processes that are required to implement suggestions; how do I handle advice?

SEER attempts to provide tentative answers to questions like these. A language, GK, is provided for programming visual knowledge. The structure of the GK encourages the user to make use of active knowledge, to think in terms of questions like: if we have x, how does that help us, can it help us get y, is there an easy way to get y given x? The control structure processes the GK representation of relevance relationships and applies them to a scene, implementing active knowledge. SEER contains the basic atomic elements, notably exploitation, that one would want for an active knowledge based system. I have analyzed some of the functions, like suggestion and advice, that such a system should carry out, and provided a specific implementation as a concrete basis for further work [4],

In short you should be able to come to this work with the idea: "I want to attack my problem with active knowledge." The response should be: "Fine. Here is how to organize your knowledge; give the result to a system with the elements of SEER, and the knowledge will be applied actively." It is encouraging to note that there is already in the literature a piece of vision research [18] which is good enough to give partial credit for its control principles to an early version of SEER [3].

SEER'S PRACTICAL ACCOMPLISHMENTS ARE QUITE LIMITED

In practical terms SEER is quite primitive. It can recognize a hammer in a few scenes. The system is extensible, but it is still a considerable effort to expand its competence. The GK is an awkward language to program in at present. An intermediate program is needed to prompt the user and "compile" simplified input formats, e.g. for loops, into the proper GK network structure. (An earlier version of SEER had a primitive input assistant of this sort.) The state of vision research is such that one still has to design and debug many basic visual mechanisms as one proceeds, e.g. for determining surface shape, particularly where one is looking for methods that can benefit from advice. Making and investigating wrong suggestions is painful, and the overhead of building and maintaining the PK data structure is significant: efficiency needs to be increased in many ways. For example, it is convenient but often wasteful to search the affinity tree in generating region arguments; too many useless regions can be conjectured about. The search process may be subject to improvement, some has already been made; but direct methods for verifying conjectured regions, like the Lozano program, should be more efficient, if they can be made as tolerant of nonuniformity as the affinity process is. Indeed it may prove that we need to be more definite about the organization of the scene into primitives units before beginning to make conjectures. Of course, given the difficulty of the visual processing problem, and the non-specialized nature of our computational hardware, we can expect any reasonably general system to be rather demanding of time and space.

SEER is limited to hammers at the moment. Obviously the basic elements of the system, the control structures, data structures, region generation, etc., are quite general, and GK knowledge

could be added for other objects. The system already deals with many different objects and properties, in the parts and features of hammers. There are issues that would be raised more forcefully by a more diverse and a larger domain. However, SEER has chosen to concentrate on those problems raised by the diversity inherent in different realistic examples of even a single class of object. SEER was tested on several scenes with real everyday hammers, with a couple of orientation, background and lighting combinations, and a case of occlusion; as opposed, for example, to several objects, a single stylized model of each, uniformly painted, and on a single contrasting background.

SEER EXTENDS AND IMPLEMENTS SEVERAL BASIC A.I. PRINCIPLES

The immediate impetus for SEER was the heterarchy concept of Minsky and Papert I13, 15, 14, 16], and its implementations in the M.I.T. "copy demo" system of Winston, Horn, Binford, Freuder, et al. [22[. Earlier ideas in domain directed processing [2, 19] also can be seen in the active knowledge approach. The basic implementation mechanisms of exploitation and exploration reflect the antecedent/consequent distinction most effectively analyzed by Hewitt [7]. The development, or at least the description, of SEER has been influenced by several ideas that came to prominence during the progress of this work. The little man or actor model (Papert, Hewitt, Kay, etc. [17, 91) provides a good metaphor for the local activity centered in the knowledge structure nodes; Minsky's frame model [12] can be related to the larger organization of the knowledge. Related work on heterarchical and knowledge-based control structures can be found in vision [11, 24, 20, 6] as well as other fields, notably speech recognition [1].

REFERENCES

[1] L. Erman & V. Lesser, "A multi-level organization for problem solving using many, diverse, cooperating sources of knowledge, IJCAI-75, 1975.

[2] H.A. Ernst, "MH-1, a computer-operated mechanical hand," D.Sc. Thesis, M.I.T., 1961.

[3] E.C. Freuder, "Active knowledge," WP-53, AI Lab, M.I.T., MA. 02139, 1973.

[4] E.C. Freuder, "A computer system for visual recognition using active knowledge," AI TR-345, AI Lab, M.I.T., MA 02139, 1976.

[5] E.C. Freuder, "Affinity: a relative approach to region growing," Comp. Graph. & Im. Proc. 5, 254-264, 1976.

[6] T. Garvey, "Perceptual strategies for purposive vision," TN-117, SRI, Menlo Pk, CA, 1976.

[7] C. Hewitt, "Description and theoretical analysis (using schemata) of PLANNER," AI TR-258, AI Lab, M.I.T., MA 02139, 1972.

[8] C. Hewitt, P. Bishop & R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," IJCAI-73,235-245, 1973.

[9] C. Hewitt & B. Smith, "Towards a programming apprentice," IEEE Trans, on S.E., SE-1, 26-45,1975

[10] T. Lozano-Perez, "Parsing intensity profiles," Comp. Graph. & Im. Proc, 6, 43-60, 1977.

[11] D. Milner, "A hierarchical picture interpretation system utilising contextual information," Dept. of Mach. Int. & Perception,U. of Edinburgh, 1970.

[12] M. Minsky, "A framework for representing knowledge," in P. Winston, Ed., The Psychology of Computer Vision, McGraw-Hill, NY, 1975.

[13] M. Minsky & S. Papert, "Research on intelligent automata," in Project MAC Progress Report IV, M.I.T. Press, MA 02139, 1967.

[14] M. Minsky & S. Papert, "Proposal to ARPA for research on artificial intelligence at M.I.T. 1970-1971," AIM 185, AI Lab, M.I.T., MA 02139 1970.

[15] M. Minsky & S. Papert, "1968-1969 progress report," AIM 200, AI Lab, M.I.T, MA 02139, 1970.

[16] M. Minsky & S. Papert, "Progress report," AIM 252, AI Lab, M.I.T., MA 02139, 1972.

[17] S. Papert, "Teaching children to be mathematicians versus teaching about mathematics," Int. J.Math. Educ. Sci. Technol., 3, 249-262, 1972.

[18] R.J. Popplestone, CM. Brown, A.P. Ambler, G.F. Crawford, "Forming models of plane-and-cylinder faceted bodies from light stripes," IJCAI-75, 664-668, 1975.

[19] H.A. Simon, The Sciences of the Artificial, M.I.T. Press, M.I.T., MA 02139, 1969.

[20] K. Turner, "Computer perception of curved objects using a tv camera, Ph.D. Thesis, U. of Edinburgh, 1974.

[21] D. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," AI TR-271, AI Lab, M.I.T., MA 02139, 1972,

[22] P.H. Winston, "The M.I.T. robot," in B. Meltzer and D. MIchie, Eds., Machine Intelligence 7, Wiley, NY, 1972.

[23] P.H. Winston, "Wizard," WP-24, AI Lab, M.I.T., MA 02139, 1972.

[24] Y. Yakimovsky and J. Feldman, "A semantics-based decision theory region analyzer," IJCAI-73, 1973.