BORIS F. KIM, JOSEPH BOHANDY, and VINCENT G. SIGILLITO

A HIERARCHICAL COMPUTER VISION PROGRAM

A computer system for the classification of simple image shapes is described that uses learning to achieve the capability to interpret simple images. It has a hierarchically distributed memory that leads to interesting consequences for the amount of memory required to interpret a number of images.

INTRODUCTION

During the course of human history, man has built machines that have capabilities far exceeding his own physical ones. Now, the digital computer gives him the potential to build systems that equal or exceed his mental abilities. Indeed, this has already proven to be the case for a few specific, highly structured tasks such as symbolic mathematical manipulations, the elucidation of chemical structure, and the diagnosis of certain types of infectious diseases. Yet many of the tasks that man performs so effortlessly—visual processing, natural language understanding, speech interpretation—have been well beyond the computer's capability, at least in anything other than very limited and circumscribed domains. Nevertheless, demands from diverse users provide a strong impetus to persevere in the difficult task of providing general, robust machine capabilities in these areas. At APL's Milton S. Eisenhower Research Center, work is being done in one of these fields—computer vision.

A vision system can be conceptualized as consisting of three components: an image acquisition component, a visual memory, and an inferential component that uses the visual memory to produce an interpretation of the image. Although the image acquisition component is generally straightforward in design and implementation, it has been exceedingly difficult to synthesize systems of memory and inference that result in useful machine vision systems. There has been a broad range of approaches to the solutions of these problems, ranging from simple template matching to mainstream artificial intelligence techniques that use knowledge bases to represent contextual knowledge. Despite over 25 years of research in machine vision, however, no machine vision system has been devised that can approach the capabilities of the human vision system.

Although some work has been directed toward developing an architecture for a general vision system, no system has been successfully implemented.[1-3] Indeed, most work in computer vision has been concerned with engineering specialized vision systems that are capable of interpreting a limited class of images, sometimes with high performance (e.g., see Ref. 4). This type of vision system often uses rules to interpret particular types of images. The rules are tailored by the designer for the specific class of images that the system is built to interpret. Such systems are of a specialized nature because of the difficulty in devising a finite set of rules that would be applicable to any image. APL's work is directed toward developing concepts we believe to be important in general-purpose machine vision systems that can interpret images from a general environment. Biological vision systems are examples of general-purpose vision systems and are the types of systems that this work seeks to emulate.

Our philosophical approach to the machine vision problem is to attempt to model, phenomenologically, certain concepts of biological vision that are understood and are thought to be important to the structure of these systems. One of our central concepts is that biological vision systems (particularly human systems) acquire the ability to interpret or recognize images of certain objects or shapes through experience. Thus, a particular object that has not been seen previously is not recognized, but it can be recognized in future encounters. This adaptive nature of biological vision systems stands in contrast to some machine vision systems in which the ability to recognize certain objects or shapes is built into the system. Instead of trying to create software algorithms to interpret particular types of images, our approach is to try to create an architecture that can learn to interpret images from experience, a concept that has been emphasized by other workers.[5]

A general-purpose vision system, of course, would have capabilities that extend beyond the recognition of simple objects. It would be able to interpret complex images containing many objects (some possibly occluded) of various sizes and orientations, and under many different lighting conditions. Our approach is to assume that this type of system can be decomposed into a number of smaller modules, each of which performs a particular task. The way in which the decomposition might occur would not necessarily be unique, and there would undoubtedly be differences of opinion concerning the best way to do this. However, one particular module, that of low-level vision (the recognition of simple objects or patterns), has been established as a significant object of study by workers in the field. Another important module would address the problem of composition (i.e., the relationship of many different objects in an image). Our work has been directed toward developing a module for low-level vision.

A MULTILEVEL RECOGNITION PARADIGM

An image classifier system has been devised that has the following characteristics:

1. The system learns to recognize image shapes or objects from experience.
2. The speed of recognition of particular images depends on when the image was last encountered; images of objects encountered most recently are more quickly recognized.
3. The system may "forget" certain images if they are not reinforced by usage.
4. The performance of the system can be increased simply by increasing the size of its image memory.
5. The system is an inherently parallel one, although the simulation described below is serial.

In the system, which has been simulated in a program called RECOGNIZE, using the LISP programming language, a hierarchy of symbolic images of monotonically decreasing size is generated, with the last image yielding an interpretation of the original image. At each level, the symbolic image is interpreted by a comparison with the elements of a memory associated with that level, and a new image is generated for interpretation recursively at the next level (Fig. 1). At a given level, the image is partitioned into an array of subarrays. Each subarray in the partitioned image is then compared with the elements in the memory for that level by a matching function. A unique symbol, which we call the "name" of the memory element, is associated with each memory element. If a subarray matches a particular memory element, the name of the element becomes the "value" of that subarray. If no memory element is found that matches the subarray, a memory element is created that matches the subarray and it is placed in the memory. A new symbol is assigned to the name of the new memory element and also as the value of the subarray.

In Fig. 1, the subarray indicated has been given the value S1. Each subarray is interpreted in this manner, the result being a new image whose pixel elements are the values assigned to the subarrays. The new image is then interpreted recursively at the next level. At the last level, the name of each memory element is a description or name of an object or pattern in the original image (first level) that was to be recognized. If the image matches a memory element at that level, the image has been seen previously and recognized, and the name of the element is displayed on a terminal screen. If no match is found, the system asks what the object is. A memory element that matches the image is stored in the memory, and a description entered from the keyboard becomes the "name" of that element. The object will then be recognized in future encounters.

The hierarchically distributed memory consists of a stack at each level that is compared sequentially with subarrays at that level. Figure 2 shows a specific example of how the symbol S1 might be created from a subarray at the top level of a binary image. Each memory element in the stack is a list of symbols of a length equal to the number of pixels in each subarray. The matching function converts the subarrays into lists that are compared directly with the memory elements as indicated. When a match is found, the memory element that was successfully matched is placed at the top of the stack, and the name of the memory element is assigned to the subarray. If no match is achieved, a memory element is created that matches the subarray. That element is placed at the top of the stack while the last element of the stack is discarded. This leads to the characteristic that the system most

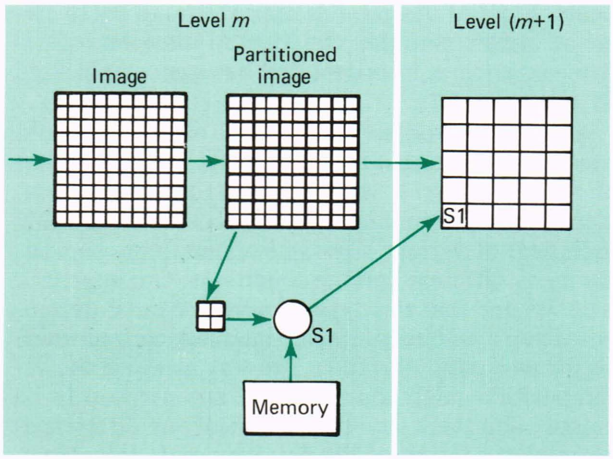

Figure 1—Two hierarchical levels in RECOGNIZE. The image in level *m* is partitioned into an array of subarrays, each of which is compared with memory. A symbol, S1, corresponding to the result of memory comparison, becomes a pixel for a new image in level *m* + 1.

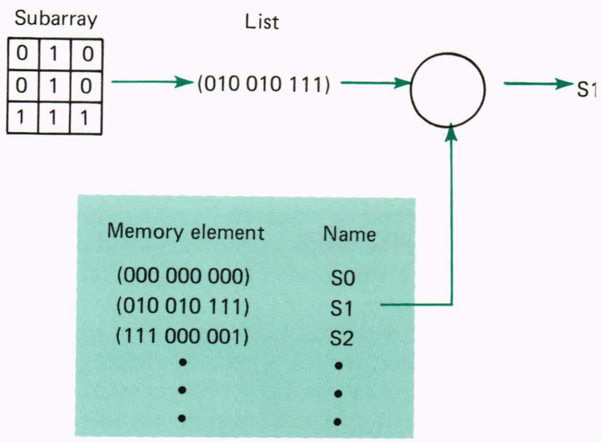

Figure 2—A comparison of a subarray with memory. The subarray is converted to a list and is compared with (memory element) lists of equal length in memory. A symbol (name) associated with the memory element that matches the list becomes a pixel element.

quickly recognizes objects it has seen most recently, and may forget or lose its ability to recognize objects that are not reinforced by usage. This is also thought to be a characteristic, to some extent, of biological vision systems. Although Fig. 2 indicates an exact match (essentially a template match) between a subarray and memory element, an exact match is not always required. The degree of the exactness of matching may be specified in each case.

The structure of RECOGNIZE is flexible and allows for a great deal of study in its use in a vision system. The parameters that apply this program to a specific image configuration are the size (number of pixels) of the image field, the number of hierarchical levels, the partition structures of the images into subarrays at each level, and the size and structure of the hierarchical memory. These parameters are specified outside the source text of the program as global variables, leading to the characteristic that performance is, to a large extent, dependent on these details, particularly the memory size.

DISCUSSION

In a general sense, the system has a common basis with some aspects of mainstream machine vision research. One aspect in particular concerns the use of "features" in image classification schemes. In typical systems that use features, the early processing seeks to detect the presence of certain features in the image and their relationships with each other. That information is then used by a knowledge-based inferential system to infer the presence of a particular object or pattern in the image. For example, one might use vertices and lines as features to detect the presence of polygons. Thus, four vertices, each connected to two neighboring vertices by lines, no two of which cross, would indicate the presence of a four-sided polygon. The idea of using features is helpful primarily if a relatively small number of features can serve to classify a large number of images. A feature might, therefore, be defined as a pattern or segment that occurs with great frequency in the entire class of images to which the system might be exposed.

From this point of view, we may consider the top few levels in RECOGNIZE as being feature detectors. There is a significant difference, however, between RECOGNIZE and most other feature-based image classifiers. In RECOGNIZE, the feature definitions are created by the program itself; in most other systems, the feature definitions are created by human designers. The lower levels of RECOGNIZE may be considered to be a knowledge-based inferential system that produces an image classification based on relationships between features detected in the top levels. As in the case of features, the knowledge base is created by RECOGNIZE, in contrast to most other feature-based systems in which the knowledge base is "handcrafted" by the designers. We believe that this characteristic of RECOGNIZE is important for the development of general-purpose machine vision systems.

One of our important concerns is the amount of memory required to interpret a given number of images. (The memory we refer to is the actual machine memory used to store image information. It does not include computer memory used to express the vision paradigm.) A useful way to assess the memory requirements of a system is to compare its requirements with those of a system that merely stores entire images in memory. In general, we expect that RECOGNIZE will require less memory than such a system when interpreting images of the physical world. Futhermore, the required average memory size per image should decrease as the number of images that can be interpreted increases. The reason for the saving in memory for a given image is that the content of some of the subarrays within the partitioned image at a given level can occur more than once. In such cases, a single memory element will serve all such subarrays. The additional saving in memory when many images have been exposed to RECOGNIZE results because subarray patterns in the various levels may occur in different images, thereby eliminating the need for an additional memory element to service those subarrays. (Patterns that recur with high frequency in the initial levels are considered to be features, as was discussed previously.)

To illustrate this point, we have applied RECOGNIZE to a particular font alphabet in capital letters. The letter "A" of the alphabet is shown in Fig. 3. The image size was 48 by 48 pixels for each letter. RECOGNIZE was arbitrarily configured to partition the image into subarrays of size 3 by 3 in the first level, 4 by 4 in the second level, and 4 by 4 in the third level. An exact match was required at each level when comparing subarrays with memory. Two cases are considered. In the first case, RECOGNIZE is applied to the 26 images with only one image in storage at a time. In the second, RECOGNIZE is applied sequentially to the 26 images so that all the images are in storage. The average number of bits of memory required per image in each case is as follows. A 48 by 48 image requires memory with the number of bits equal to the number of pixels (48^2), or 2304. RECOGNIZE, with one image at a time, requires 1549 bits. But RECOGNIZE, applied sequentially to the 26 images, requires only 659 bits. The results illustrate the point discussed in the preceding paragraph and give some indication of the merit of RECOGNIZE with respect to memory requirements.

Another point of interest that has not been investigated fully is the use of partial matching in the interpretation of images with the system. It would seem that partial matching would be desirable for a number of reasons. First, noise in the image would be less detrimental. Noisy images would prevent proper image classification in systems requiring an exact match. Another advantage is that the system should be able to interpret a domain of images that is larger than its domain of previous experience. (For example, the system, having learned to recognize the alphabet of a particular font, might be able to recognize characters from another font.) Finally, we expect that the average mem-

```
                 1
                 1
                1 1
                1 1
                1 1
               1 1 1
               1 1 1
              1 1 11
              1 1 1 1
              1 1 1 1
             1   1 11
             1   1 11
             1   1 1 1
            1     11 1
            1     1 11
            1     1 1 1
           1       11 1
           1       1 11
          1        1 1 1
          1         11 1
          1         1 11
         1          1 1 1
         1           11 1
         1           11 1
        1            1 1 1
        1             11 1
        1             11 1
       111111111111111111 1
       1               11 1
       1               11 1
      1                1 11
      1                 11 1
     1                  11 1
     1                  11 1
     1                   1111
    1                    11 1
    1                    11 11
   1 11                 1111 11
111111111111        11111111111111
```

Figure 3—The letter A in the alphabet used to test RECOGNIZE.

ory capacity required per image should be less when partial matching is used. Of course, there will also be some decrease in accuracy in image interpretation. The degree to which a system such as RECOGNIZE is affected by using partial matching in its hierarchical image comparisons will depend on the matching criteria and how these criteria are applied in the different levels of the hierarchy.

CONCLUSION

The system described here has some characteristics that are similar to human vision, the principal one being that its ability to interpret objects or patterns is based on its prior experience. This is in contrast to vision systems that use various specific modeling techniques to interpret specific classes of images. RECOGNIZE is general in the sense that it is not restricted in the types of patterns or objects it can recognize. It can work with binary or gray scale and with color images. Execution times are generally slow because the paradigm is expressed serially in a LISP environment. The structure of the paradigm is inherently parallel, however, and it might be implemented by a highly parallel pipeline architecture of cellular arrays.[6]

In principle, RECOGNIZE could be part of a machine vision system. A complete system would, of course, require more components, both at lower and higher levels of processing. At lower levels, for example, processing is needed to present normalized images to an image classifier such as RECOGNIZE, which does not account for variations in size and orientation of patterns. Higher levels of processing are required to interpret scenes that consist of multiple objects with arbitrary backgrounds and to account for other aspects of vision such as motion. Further work on the project will be directed toward the problem of accounting for variations in size and orientation.

REFERENCES

[1] D. Marr, *Vision,* W. H. Freeman and Co., New York (1982).

[2] H. G. Barrow and J. M. Tenenbaum, "Computational Vision," *Proc. IEEE* **69**, 572-595 (1981).

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

[4] R. A. Brooks, R. Greiner, and T. O. Binford, "The ACRONYM Model-Based Vision System," *Proc. IJCAI* **6**, Tokyo, 105-113 (1979).

[5] R. M. Haralick and J. R. Ullmann, "A Report on the 1982 IEEE Computer Vision Workshop," in *Proc. Workshop on Computer Vision, Representation and Control,* pp. VII-XIII (1984).

[6] A. Rosenfeld, "Parallel Image Processing Using Cellular Arrays," *IEEE Comput. Mag.,* 14-20 (1983).

ACKNOWLEDGMENT—We wish to thank R. E. Sterner for providing us with digital images. The assistance of R. F. Wachter is also gratefully acknowledged for writing a program to format the images.

THE AUTHORS

BORIS F. KIM (right) was born in Ashland, Ga., in 1938. He received a B.E.S. degree in electrical engineering in 1960 and a Ph.D. in physics in 1967 from The Johns Hopkins University. Dr. Kim served as an officer in the U.S. Army during 1967-69, after which he joined APL. His research interests are in the fields of experimental atomic and molecular spectroscopy, and computer vision.

JOSEPH BOHANDY (left), a native of Ohio, received B.S., M.S., and Ph.D. degrees from Ohio State University. He joined APL in 1965 as a member of the Microwave Physics Group in the Milton S. Eisenhower Research Center and worked on various problems in solid-state and chemical physics. In 1984, he became a member of the Materials Science Group in the Research Center, where his current professional interests include both computer vision and the photochemistry and photophysics of processes related to microcircuit fabrication.

VINCENT G. SIGILLITO's biography can be found on p. 18.