A Neural Network Architecture for General Image Recognition

Robert L. Harvey, Paul N. DiCaprio, and Karl G. Heinemann

As part of Lincoln Laboratory's research on neural network technology, a general-purpose machine vision system that can learn to recognize diverse objects has been designed. The system models human vision, primarily with neural networks, and the learning is by example.

We tested the system on two disparate classes of objects—military vehicles and human cells—with video images of natural scenes. These objects were chosen because large databases were available and because most researchers judged the two types of objects unrelated. When we trained and tested the system on 40 images of military vehicles, the system was able to recognize the tanks, howitzers, and armored personnel carriers without any errors. Pathologists at Lahey Clinic Medical Center collaborated in the cytology study in which we trained and tested the system on 156 cell images from human cervical Pap smears. The system recognized normal and abnormal (i.e., precancer) cells perfectly. Because of the small number of samples of the military vehicles and Pap-smear cells, these results are preliminary.

We should note that the architecture of the system is applicable to many civilian and military tasks. The application depends mainly on training.

If one way be better than another, that you may be sure is Nature's way.

—Aristotle

RESEARCHERS HAVE BEEN STUDYING machine vision (MV) technology for more than 30 years. As a result of their work, a standard technology for computer vision has evolved. The most rigorous of the conventional MV methods comes from D. Marr's work at MIT in the 1970s (see the box "Conventional Machine Vision Design"). Yet, to some researchers and potential users, the performance of conventional MV systems is disappointing. To establish the context of our work, we quote from a recent review by A. Rosenfeld [1], a founder and leading MV figure:

> . . . Standard vision techniques for feature detection, segmentation, recovery, etc., often do not perform very well when applied to natural scenes.
>
> Ideally, the [vision process] stages should be closely integrated; the results obtained at a given stage should provide feedback to modify the techniques used at previous stages. This is rarely done in existing vision systems, and as a result, little is known about how to design systems that incorporate feedback between stages.
>
> . . . Humans can recognize objects—even complex objects whose presence was unexpected—in a fraction of a second, which is enough time for only a few hundred (!) "cycles" of the neural "hardware" in the human visual system. Computer vision systems have a long way to go before they will be able to match this performance.

Our main goal was to develop a general MV architecture that would work on a variety of image types without significant changes in the algorithm. This robustness contrasts with the current practice of tailoring an MV system to a specific application. Such tailored systems have not performed well in situations unforeseen by the designers.

In many respects the human vision system—known for its high performance over a wide range of objects and situations—is far superior to current MV systems. Thus, in developing the architecture of our system, we decided to model the human vision system. Although this idea

CONVENTIONAL MACHINE VISION DESIGN

TO A MAJOR EXTENT, conventional machine vision (MV) technology evolved from the work of one man, D. Marr, at MIT in the 1970s. Developed from the information theory, cybernetics, and digital computer technology of that era, Marr's contributions were made within the field of artificial intelligence. His work led to numerous papers on vision and, finally, to the book *Vision* [1], which was published after Marr died of leukemia in 1980 at the age of 35. Reference 2 provides a recent summary of Marr's work in relation to that of others.

An acknowledged contribution of Marr's was his attempt to clarify the thinking about vision systems or, more generally, information processing systems. In his work, Marr introduced the distinction among three levels of explanations: (1) the computational theory, (2) the algorithm, and (3) the hardware implementation. Consideration of these levels and associated issues leads to a sequence of questions that guides the design.

At the time, Marr's explicit ideas somewhat puzzled researchers in vision because other approaches used concepts that were undefined, or they used descriptions rather than explanations. Marr attained a high degree of rigor because his approach produced ideas that could be directly checked by computer simulation.

At the computational-theory level, Marr asserted that the key issue is to determine both a goal for the computation and strategies for achieving that goal. By explicitly stating the goal and accompanying strategies, we can describe what the machine achieves and characterize the constraints. Knowledge of the constraints, in turn, allows the processes to be defined. At the algorithm level, the key issue is how the input and output are represented, and the actual algorithm for transformation. The algorithm will depend partially on the nature of the data representation. At the implementation level, the key issue is how the machine actually works. The concern here is with the hardware of the machine, and the nature and operation of its component parts.

For example, applying Marr's approach to optical sensors and two-dimensional processing leads to a modular design with the following consecutive processing stages:

Stage 1. Extract features such as edges from an image to produce a map representation. The map (called the primal sketch) consists of pixels and their feature values, such as edge strengths. (In this context, edge strengths are various combinations of first and second derivatives at each point in the image.)

Stage 2. Improve the map by grouping pixels in connected regions.

Stage 3. Represent the map by an abstract relational structure.

Stage 4. Recognize objects by comparing the structure with stored models.

Three-dimensional scenes are an extension of two-dimensional ones. For the extension to three dimensions, stages 1 and 2 should be replaced by a method to find the surface orientation of each pixel. The process will produce a representation map called the 2½-D sketch.

Further extensions of Marr's method add one or more of the following stages: (1) cleanup of input pixel values with image-restoration techniques, (2) production of multiple images for stereomapping and motion analysis, (3) adjustment of the processing by feedback from later stages to earlier stages, and (4) recognition of objects by matching them with models made up of composite parts.

References

1. D. Marr, *Vision* (W.H. Freeman, San Francisco, 1982).
2. I. Gordon, "Marr's Computational Approach to Visual Perception," in *Theories of Visual Perception* (John Wiley, New York, 1989), chap. 8.

was not new, our approach was; it was to model the entire system module by module, primarily through the use of neural networks. We also incorporated features that give the human vision system advantages over MV systems, namely, feedback, parallel architecture, and a flexible control structure. We tested the system module by module and in its major composite subsystems.

Two new technologies made this study feasible: neural network theory and a new class of computers. These technologies were not available in the 1970s, when Marr developed his method.

The Approach—A Biologically Inspired System

Our approach was to model the human brain. Thus we based the functions and names of different modules on structures in the human vision system (see the box "Human Vision—A Brief Summary"). The modules roughly approximated functions of their biological counterparts. For convenience, we used a mixture of neural networks and standard processing algorithms to implement the module functions.

Our system recognizes gray images in a field of view (FOV); the images can have arbitrary translations and rotations. We omitted certain biological features—binocularity, size invariance, motion perception, color sensitivity, and the discernment of virtual boundaries—because we wanted to study the simplest architecture. Moreover, these features are unnecessary for many appli-

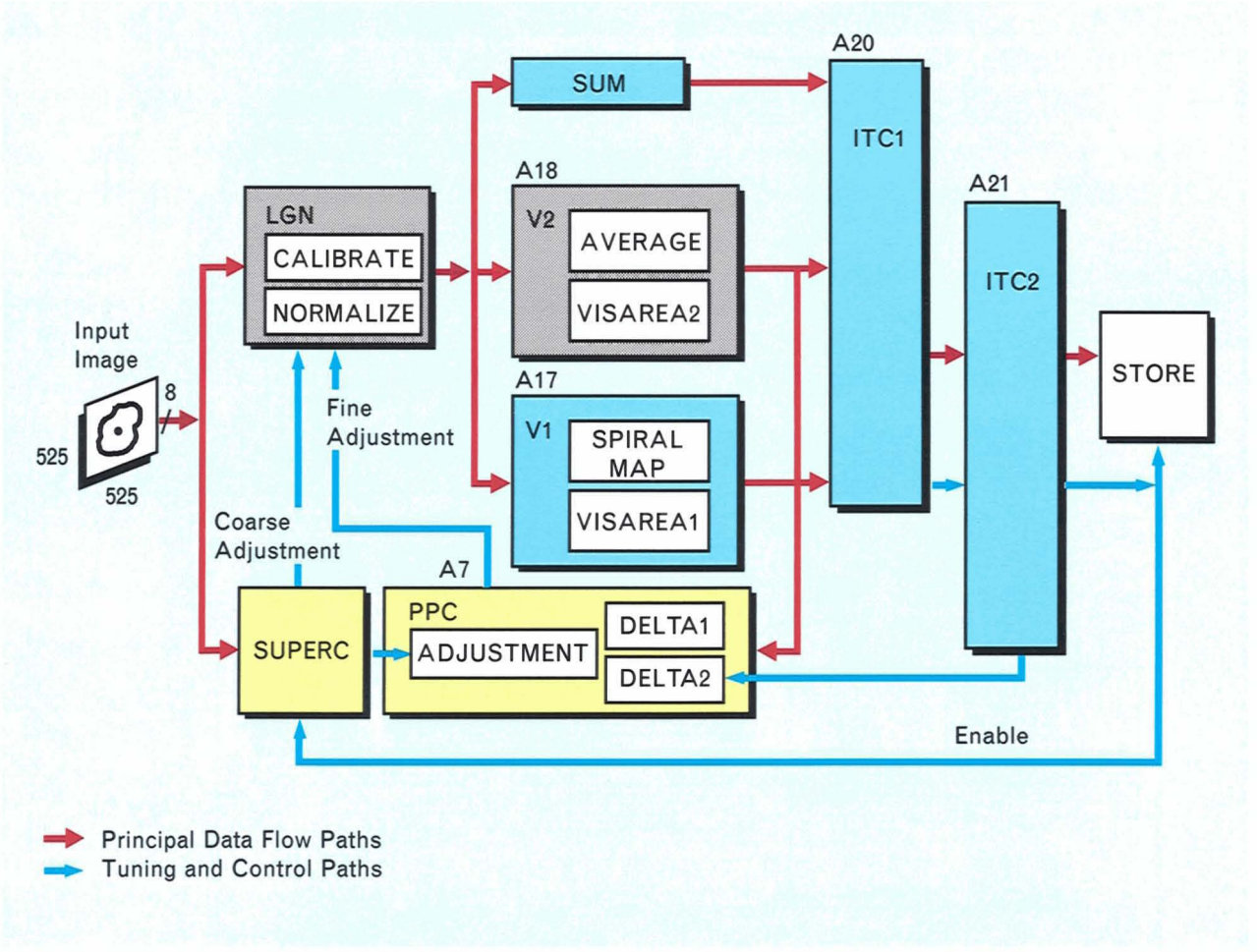


FIGURE 1. Block diagram of the neural network architecture. The modules that are in the location channel are shown shaded in yellow, the modules in the classification channel are shaded in blue, and the modules common to both channels are shaded in gray. Some modules are neural networks; others use conventional processing (see text). A 525 × 525 image with 8-bit pixels has been included at the left of the figure as an example input image. (Note: See the box "Glossary of Acronyms" for definitions of the acronyms used.)

HUMAN VISION— A BRIEF SUMMARY

THE HUMAN BRAIN has three main structures—the forebrain, the midbrain, and the hindbrain. Figure A (top) shows the forebrain, or neocortex. Red indicates some of the areas associated with vision.

The visual cortex occupies the entire back half of the neocortex hemisphere. At least a dozen cortical areas (not all shown in Figure A but described by D.H. Hubel in Reference 1) are involved with the vision process. Areas 17, 18, and 19 are feature detectors, areas 20 and 21 function as classifiers, and area 7b helps to locate objects in the field of view (FOV). (Note: The numbering scheme was originated

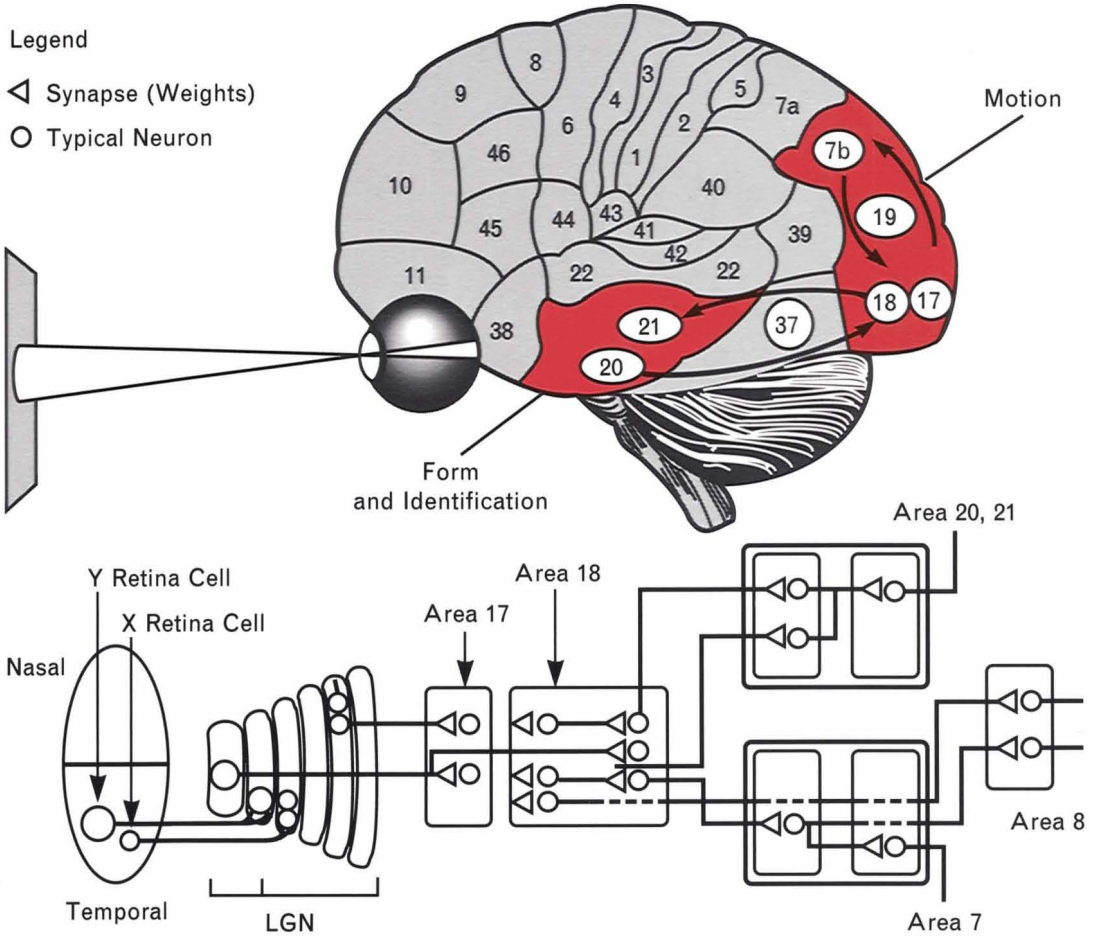


FIGURE A. Neural network vision model: (top) sketch of human forebrain in which areas associated with vision are shown shaded in red, and (bottom) block diagram of human vision process.

cations (see the section "Applications and Extensions").

The architecture of the system includes two major channels that work together. The location channel searches for objects of interest in the FOV and, after one is found, the classification channel classifies it. Studies of the human vision system as well as that of other animals suggest that the locating and classifying functions are separate [2].

by K. Brodmann in 1909.)

Certain modules in the midbrain also belong to the visual system: for example, the lateral geniculate nucleus (LGN), shown in Figure A (bottom). The optic nerves relay the images captured by the eyeballs to the LGN, which has processing functions and acts as a buffer.

The retina has about 1.25×10^8 receptors. Data compression by retinal processing is about 125 to 1, which gives a resolution near the fovea of about 1000×1000 pixels. Assuming 7 bits of relative discrimination of stimulus frequency per pixel [2] and a 100-Hz pulse frequency along the optic nerve, we find that the data rate to the visual cortex is about 700 Mb/sec, less than the capacity of fiber optic channels.

Researchers have mapped over 30 pathways among the visual areas but the actual number probably is much larger because there are many connections to areas that have not been studied. A basic finding is that with few exceptions the pathways connect the modules in reciprocal fashion.

Evidence shows a hierarchical structure for the vision system [3]—for the dozen visual areas, the overall cortical hierarchy has six levels. Anatomical, behavioral, and physiological data show two distinct channels for classifying and locating. In Figure A (bottom), the classification channel consists of the LGN, A17, A18, A19 (not shown in the figure), A20, and A21; the location channel consists of the LGN, A18, A7, and A8.

The two channels separate at the retina and they have their own retinal cells, labeled X and Y. (Another cell type not shown, the W cell, goes to the midbrain areas to coordinate the FOV to head and eye movements). At the cortical and midbrain levels, the channels remain separate. Evidence shows the classification channel analyzes form and color while the location channel analyzes visual motion across the FOV.

In A17, A18, and A19, three types of cells work like feature detectors. The features in primate vision are stationary or moving edges, slots or lines, and their respective ends [1]. (In comparison, the features in current MV technology are much more varied: corners, human faces, spatial frequencies, and responses of matched filters are typically used).

The population of retinal cells that feed into a given feature cell are not scattered about all over the retina but are clustered in a small area. This area of the retina is called the receptive field of the feature cell. The size of the receptive field of simple cells is about one-quarter degree by one-quarter degree at the fovea.

Research shows that feedback takes place in the human vision system [4]. In normal operation, stimulating A20 changes the receptive fields of A17. This result suggests that A20 exerts feedback control over the feature detectors. Thus recognition is likely an active feedback process that restructures the feature-extraction stage. The restructuring continues until the transformed input matches some known class of stimulus. Research also suggests that there is a mechanism for directing attention within the FOV. In short, *windowing* takes place. Windowing focuses attention on small details and also suppresses notice of other objects in the FOV. Researchers suspect that the midbrain directs the windowing process automatically, perhaps by using cortical inputs.

In summary, human vision is a system in which a small number of serial stages (sensor-preprocessor-feature-extracting-classifier) process large arrays of data in parallel. The architecture has two channels that use feedforward and feedback signals.

References

1. D.H. Hubel, *Eye, Brain, and Vision* (W.H. Freeman, New York, 1988).
2. T.B. Sheridan and W.R. Ferrell, *Man-Machine Systems* (MIT Press, Cambridge, 1974), p. 261.
3. D.C. Van Essen and J.H.R. Maunsell, "Hierarchical Organization and Functional Streams in the Visual Cortex," *Trends Neurosci.* **6**, 370 (1983).
4. E.W. Kent, *The Brains of Man and Machines* (McGraw-Hill, New York, 1981).

Figure 1 shows a block diagram of the architecture, the box "Glossary of Acronyms" contains a list of the acronyms used, and the following sections briefly describe the functions of each module. We used feedforward and feedback paths to coordinate the modules. To illustrate how the system works, we carry through an example of a 525×525-pixel input image with 8-bit pixels. For convenience, we start our description with

the classification channel and the modules that are common to both channels.

Classification Channel

Certain classification-channel modules approximate the functioning of selected brain areas: the lateral geniculate nucleus (LGN), visual area 1 (V1, also called A17; see the box "Human Vision—A Brief Summary"), visual area 2 (V2, also called A18), inferior temporal cortex 1 (ITC1, also called A20), and inferior temporal cortex 2 (ITC2, also called A21). Other modules, such as the SUM module, approximate certain biological functions without the anatomical correspondence, as described below.

GLOSSARY OF ACRONYMS

FOV—field of view

ITC1—inferior temporal cortex 1 (also called A20); an unsupervised classifier

ITC2—inferior temporal cortex 2 (also called A21); a supervised classifier

LGN—lateral geniculate nucleus; buffers and places a window around an object

LTM—long-term memory

MV—machine vision

PPC—posterior parietal cortex; centers a window about an object

SUPERC—superior colliculus; performs coarse location of an object

V1—visual area 1 (also called A17); extracts high-resolution features from an image

V2—visual area 2 (also called A18); extracts information about an object's general shape

LGN—Grayness Processing

Figure 2 shows the front-end processing in the classification channel. The LGN uses inputs from the location channel (see the following section) to place a window around an object. We set the window size to fit the object, so the system does not emulate size invariance. Starting with a 525 × 525-pixel input image, our example uses a window of 175 × 175 pixels.

The CALIBRATE and NORMALIZE boxes of LGN (Figure 1) operate on gray-scale imagery. CALIBRATE performs a histogram equalization of the object's pixel values, and NORMALIZE rescales the pixel values so that they leave LGN within a range from zero to one. These two procedures enhance the image contrast and adjust for varying brightness in the FOV.

V1—High-Resolution Features

The first feature-generating module is V1, which breaks the input window into subwindows of 7 × 7 pixels. Thus, in our example of a 175 × 175-pixel window, there are 625 subwindows. Note that the 7 × 7 subwindow size is unrelated to the input image size. Each 7 × 7 subwindow is then processed by SPIRAL MAP and VISAREA1.

SPIRAL MAP (Figure 1) scans through the subwindows in a spiral pattern. The mapping proceeds as follows: left to right across the top row, down the right column, right to left across the bottom row, up the left column, back across the second row, and so forth until the process ends at the center subwindow. The purpose of the spiral mapping is to simplify interpretation of the feature data.

VISAREA1 (Figures 1 and 2) does the high-resolution feature extraction. For each 7 × 7-pixel subwindow, VISAREA1 measures luminance gradients (increasing or decreasing) in four different directions. A gradient is a characteristic of gray images, and is analogous to an edge in a binary image. The luminance gradient in our system is the rate of change, or slope, in brightness across a 7 × 7 subwindow. Windows with an abrupt step in brightness in one direction will have a large gradient in that direction; windows with a gradual change in brightness from one side to the other will have a small gradient; and windows with uniform brightness, i.e., windows with no visible edges, will have zero gradient.

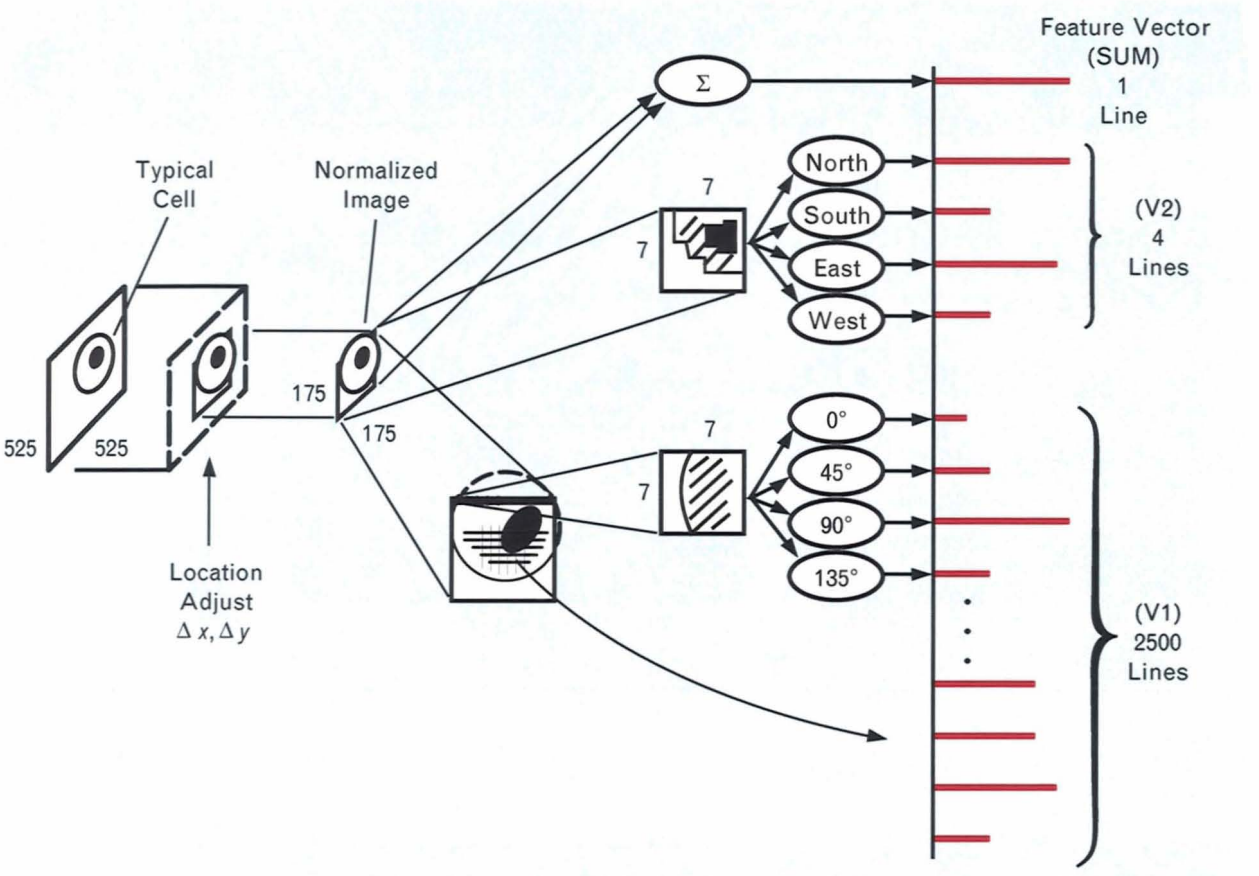


FIGURE 2. Summary of image processing operations. For the example of a 525 × 525 input image (Figure 1), the classification channel places a window of 175 × 175 pixels around the object in the image. (The window size is set to fit the object size.) The 175 × 175 window is then broken into subwindows to extract details of the image, and the details are stored in a feature vector.

Because the slope depends on direction, the gradients in different directions are usually not the same. The system produces representative gradients in four directions—vertical, horizontal, and the 45° diagonals—for each 7 × 7 subwindow.

The gradient detectors in the system use cooperative-competitive neural networks that model similar biological processes. In biological nervous systems, a neuron is either excitatory or inhibitory; i.e., it either attempts to turn on or turn off other neurons [3]. A general cooperative-competitive neural network employs a mixture of interacting excitatory and inhibitory neurons. Cooperative-competitive neural networks are one type of neural network. Other common types are special cases that can consist of only inhibitory neurons, which produce on-center/off-surround networks.

For binary images, we found that on-center/off-surround networks could detect the edges and measure their orientations. However, the gradients in gray images required cooperative-competitive networks for the same tasks. Our feature-extracting neural networks had 25 hidden neurons and one output neuron. We used fixed neuron connection weights that we computed off-line by a genetic algorithm method, described in Reference 4.

To help interpret the feature values obtained from SPIRAL MAP and VISAREA1, we arranged the V1 outputs in a vertical vector (Figure 2). For the 175 × 175-pixel window, there were 2500 V1 feature values because each 7 × 7 subwindow produced four values. We stored feature values from the image's outer parts at the top of the vector and feature values from inner parts at the bottom of the same vector. Thus data about the general shape of an image could be found at the top of the vector and data about the interior at the bottom.

V2—Shape Features

The second feature-generating module is V2 (Figures 1 and 2), which detects edges near the perimeter of the input window. V2 is also part of the location channel (see the following section), and its output contains information about an object's general shape.

To detect edges, V2 produces a single defocused 7×7 image of the 175×175 input. In the defocused image, each pixel corresponds to a subwindow of the original input. AVERAGE sets a subwindow size that partitions the input image and then computes a mean pixel value for each of the subwindows. For the 175×175-pixel window, the subwindows are 25×25 pixels.

VISAREA2 detects edges near the four sides of the defocused image. The output of this module, which uses cooperative-competitive neural networks similar to those in V1, contains four values that measure the edge strengths on the north, east, south, and west sides of the 175×175 image.

SUM—Size Feature

The third feature-generating module is SUM (Figures 1 and 2), which adds up the pixel values of the input window. Thus the single output from SUM measures the object's gross size. The corresponding biological function occurs in both V1 and V2, but we have made it separate for the sake of convenience.

Feature Vector

The system classifies objects by using features based on detailed structure (V1), overall shape (V2), and size (SUM). The different subwindow sizes of V1, V2, and SUM approximate the different size-receptive areas of the visual cortex. For the 175×175-pixel window, there are 2500 values from V1, four values from V2, and one value from SUM. These 2505 values form the feature vector. We can adjust the values of each module's output to give equal influence to an object's size, shape, and detailed structure (see the section "Test Results").

ITC1 Module—Unsupervised Classification

The recognition process consists of an unsupervised classifier (ITC1) followed by a supervised one (ITC2). For the unsupervised classifier, we used the well-known ART-2 neural network. We selected ART-2 over other neural network classifiers, such as perceptrons and Hopfield nets, because of ART-2's speed, stability, feature amplification, and noise reduction—features that were better suited to our application. ART-2 is also a better model of the biology.

Adaptive Resonance Theory (ART) is a learning theory introduced by Boston University professors G. Carpenter and S. Grossberg [5]. ART mimics the human brain by taking inputs from the environment, organizing the inputs into internally defined categories, and then recognizing similar patterns in the future.

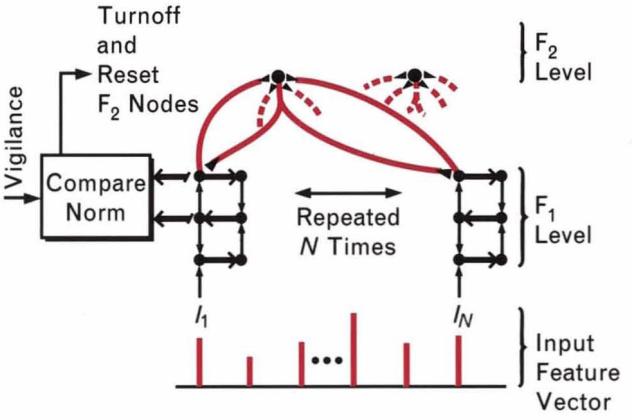


FIGURE 3. Summary of the ART-2 classifier, a neural network with two levels F_1 and F_2 that consist of interconnected neurons. The bottom layer of F_1 receives an input pattern that is then filtered, enhanced, and rescaled by the three F_1 layers. The filtered pattern appears at F_1's top layer, which is connected to the F_2 level. The filtered pattern is the pattern that ART-2 stores.

There are three classes of ARTs. ART-1, which was developed first, is used with binary inputs; ART-2 is used with patterns consisting of real numbers; and ART-3 handles sequences of asynchronous input patterns in real time. This study used ART-2. Several versions of ART-2 exist, but they all have the same basic characteristics described below.

Figure 3 shows the basic structure of ART-2, a neural network with two levels of interconnected neurons, F_1 and F_2. The neurons are mathematical models of biological neurons. In the figure, the bottom layer of F_1 receives the input pattern—a list of numbers representing the input. F_1 consists of three layers of interconnected neurons that filter out noise, enhance the shape of the pattern, and rescale the input pattern values. The filtered pattern appears at F_1's top layer, which is connected to

the F_2 level. The filtered pattern, called the exemplar, is the pattern that ART-2 stores.

In the F_2 level, each neuron represents a category, or, with high sensitivity (see below), one example input that defines a category. The activation of F_1 and F_2 models the activation of biological neurons.

The F_1 and F_2 levels are connected in both a bottom-up and top-down fashion by the long-term-memory (LTM) trace. Mathematically, the LTM trace is the set of weights given to the F_1 neurons as they attempt to turn on an F_2 node. Functionally, the LTM trace stores information permanently or until the trace is modified by learning. The LTM trace models the synaptic junctions of biological neurons.

To train an ART-2, the initial LTM trace values are set according to a rule given by Carpenter and Grossberg [5]. Next, a set of training patterns is presented to F_1 one after another. Initially, when ART-2 is untrained, the first pattern immediately causes the neural network to enter into the learning mode. The network learns the pattern by modifying the weights associated with one of the F_2 nodes.

After the first pattern is learned, each succeeding pattern will trigger the network to search for a match among the F_2 nodes. If the pattern is a close match to a previously learned pattern, ART-2 enters the learning mode and modifies the LTM trace so that the trace is a composition of all the past, closely matched patterns. If the pattern is mismatched with all the previously learned patterns, ART-2 goes into the learning mode and learns the pattern by modifying the weights associated with an unused F_2 node. Thus each pattern is automatically associated with an F_2 node, and in this manner ART-2 programs itself.

After training is completed and a new pattern is presented, the pattern's exemplar is produced, and ART-2 searches the LTM trace for the stored pattern that most closely matches the exemplar. When a match is found the corresponding F_2 neuron turns on, indicating the category that best matches the pattern.

Before using ART-2, we must set several parameters that influence the network's performance. For many of these parameters, suitable values have been determined by experience. One parameter of importance is the Vigilance, which serves as a threshold on the degree of similarity between the LTM trace and the input pattern's exemplar. If a certain mathematical matching formula equals or exceeds the Vigilance, that pattern will be associated with the corresponding F_2 node. When the Vigilance criterion is not satisfied, ART-2 declares a mismatch and searches for a match among the other nodes.

The selection of a low Vigilance value (i.e., a value near 0) leads the system to tolerate large differences, resulting in coarsely defined categories. A high Vigilance value (i.e., a value near 1) leads to increased sensitivity in pattern discrimination, resulting in finely defined categories. In practice, the Vigilance should be adjusted high enough to distinguish patterns that represent different categories. However, the value should be low enough that slight changes resulting from incomplete or wrong information will not cause mismatches.

ITC2 Module—External Labels and Flexible Control

After training, the ITC1 (ART-2) output nodes in F_2 correspond to particular patterns, or objects. For instance, if the first ten examples are tanks, the first ten ITC1 output nodes will correspond to tanks. In our basic system, the supervised classifier ITC2 uses a simple logical OR operation to associate activity of any of these nodes with the name TANK.

(Note: ITC1 is called an unsupervised classifier because the label of an input pattern is the F_2 node number, which is automatically and internally defined by the algorithm. ITC2 is called a supervised classifier because the user defines the labels externally.)

After ITC2 processing, the system decides whether to store the object's name and location, and the ART-2 matching parameter [5] serves as a confidence measure for the decision process. If the matching parameter just passes a threshold (the Vigilance), the confidence level is 50%. A perfect match corresponds to a confidence level of 100%. If the confidence level passes a second threshold specified by the user, the system stores the results. But if the confidence is not high enough, the location channel adjusts the window (discussed in the following section) and the system processes the image again.

Location Channel

The location channel places an input window around an object so that the system might classify it. As shown in Figure 1, the location channel consists of the following modules: superior colliculus (SUPERC), LGN, V2, and

posterior parietal cortex (PPC). Location is a two-stage process consisting of coarse location followed by pull-in.

SUPERC—Coarse Location

The SUPERC module uses a second ART-2 neural network to perform coarse location. The network's LTM trace, which we compute off-line, corresponds to general shapes of interest. This trace primes the system. To detect the presence of an object, the SUPERC ART-2 compares the exemplar of its current window to the LTM trace. Even an off-center object will trigger a match if the object's size is correct.

SUPERC judges a match by comparing the ART-2 matching parameter with a threshold [5]. If the system does not find a match, SUPERC shifts its attention to an abutting window. When a match does occur, SUPERC sends signals to other modules; the ART-2 matching parameter is an Enable signal for LGN, and the PPC module receives the coarse position as a starting point to center the window.

PPC—Pull-In

Pull-in operates over a feedback path that consists of LGN, V2, and PPC. Using the outputs of V2, PPC makes small changes in the window's position. When the system centers a window on an object, all the V2 edge strengths are about equal. Otherwise, PPC tries to equalize the V2 edge strengths. For example, Figure 2 shows an object that is above and to the right of the window. This position produces a stronger north than south response and a stronger east than west response because of the stronger gradient. To center the object, the DELTA-1 box (Figure 1) must move the window north and east.

A second pull-in path, which consists of LGN, V2, ITC1, ITC2, and PPC, makes repeated tries at recognition. ITC2 activates this path when the classification channel has low confidence in a match between an input pattern and the closest stored pattern. When the path is activated, the DELTA-2 box generates a small, random adjustment of the window's position and the system then tries to classify the object with greater confidence. A counter limits the number of tries.

The Software Testbed

One of our major goals was to test the architecture with computer simulation. To that end, we developed a series of software testbeds to study the algorithm performance. As we programmed the testbeds to handle more complex and extensive data, the architecture and algorithms evolved.

We built up the software in three stages. The earliest version was written in the APL*PLUS programming language and ran on an IBM PC/AT. Using synthetic binary images such as alphabetic letters, the APL*PLUS software tested algorithms for the individual modules.

We developed a second version for Sun Microsystems workstations. The Sun testbed provided a convenient operator interface, handled gray-scale images from real sensors, and incorporated algorithm modifications that were needed to process gray-scale data.

A third version incorporated a Convex C220 supercomputer along with the Sun 4/110 or SPARC workstation. The Sun/Convex testbed increased the run speed while maintaining the previous version's operator interface and options. We used the Sun/Convex testbed to test the databases of military vehicles and Pap-smear images.

Source code for the Sun and Sun/Convex testbeds was written in the C programming language, and the programs ran under the UNIX operating system. In the Sun/Convex testbed, we distributed the module functions between the two computers: the Sun workstation performed I/O and interacted with the operator while the Convex computer carried out the V1 and ART-2 calculations. A local area network enabled communication among the separate subprograms.

Adapting and optimizing the V1 and ART-2 source code for the Convex computer resulted in performance gains that were dramatic (our colleague C. Mehanian optimized the ART-2 algorithm). For example, the Sun/Convex testbed ran V1 over a 175×175 image in only 45 sec, while a Sun 4/110 required 4½ min. On the Convex, ART-2 learned the corresponding feature vector in only 18 sec, as compared to 20 min on a Sun 4/110.

Figure 4 shows the Sun screen display, which consists of a set of windows. The operator enters data through the keyboard and selects run options with a mouse. Algorithm execution can be monitored by text and graphical displays. Figure 4 displays a normal cell, its histogram, and its feature vector.

To test large databases, we have programmed a batch

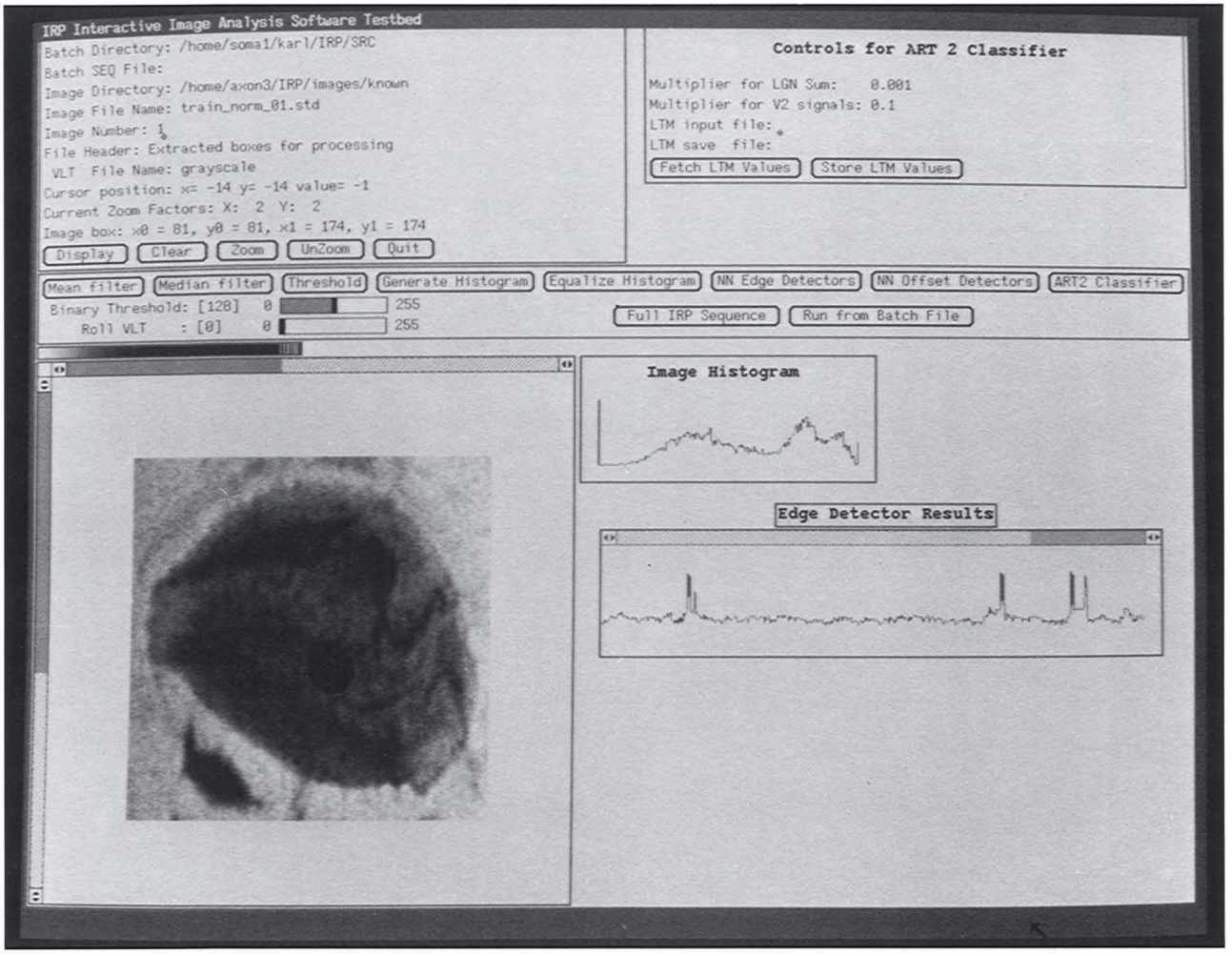


FIGURE 4. Sun workstation display of a Pap-smear cell, its histogram, and its feature vector.

mode in which a list of input images can be read automatically from a disk file. The batch option runs the images through the algorithm without operator intervention.

Test Results

We tested the major subsystems to assess their performance. To simplify the interpretation of the results, we tested the location and classification channels separately. For the classification channel, we centered, or foveated, the objects by hand so that the tests evaluated recognition under ideal conditions. Thus our results gave an upper bound on system performance because the location process introduces additional errors.

The preliminary test results were auspicious. The system located and recognized objects in their natural settings, and the algorithm was robust with respect to centering accuracy and clutter. In the following subsections we describe the classification-channel tests. (Note: The location-channel tests are contained in Reference 4.)

Military Vehicles

We assembled a dataset of three common military vehicles: an M48A5 tank, an M113 armored personnel carrier (APC), and an M110 self-propelled howitzer. The database consisted of 40 images: 20 tanks, four APCs, and 16 howitzers. The vehicles were at 700-m range, with orientations that varied from front-on to broadside to end-on, and the background consisted of trees and rolling hills.

The images, intensity measurements made with a low-level TV camera, were 120 × 128 pixels with each pixel containing eight bits, so that 2^8 gray values could be represented. We made no effort to improve the

FIGURE 5. Typical intensity image of a howitzer.

images with, for example, preprocessing. Figure 5 shows a typical image of a howitzer.

To simulate the foveation of an image, we centered a 42 × 42-pixel window by hand. The system then generated a feature vector that contained three kinds of data: SUM, V1, and V2. We varied the relative weighting among the three components. Large SUM and V2 values resulted in V1 having little effect on recognition while small SUM and V2 values allowed V1 to play a predominant role. By adjusting the SUM and V2 multipliers, we gave the three feature-vector components roughly equal influence.

To find suitable weighting values, we trained the system on a small set of images and watched the resulting number of categories that ART-2 formed. Figure 6 shows some of our results for an 18-image training set. Note that when the SUM and V2 multipliers have low values, the number of categories is the same as the number of inputs. As the SUM and V2 multipliers increase, the V1 features become less important, and the system loses its ability to discriminate between certain categories. Consequently, the number of ART-2 categories decreases below the number of input patterns. The dotted line in Figure 6 represents a rough boundary for this transition. To obtain equal weighting for SUM, V1, and V2, we chose a point inside the boundary near the elbow. In general, the SUM and V2 multiplier values depend on the window size and sensor properties.

We trained the system on 10 random tank images that spanned orientations from head-on to end-on, and the corresponding exemplars were stored in ART-2 nodes 0 through 9. Next we trained the system on eight howitzers, which were stored in ART-2 nodes 10 through 17. During training we used a Vigilance of 0.999, which corresponded to a 2.6° angular separation in feature-vector space.

After the training was completed, we tested the system on the remaining 22 images. Table 1, which summarizes the results, shows that the system correctly classified the remaining 10 tanks and eight howitzers. When the Vigilance was set to 0.997, which corresponded to an angular separation of 4.4°, the four APC images went to an untrained node. Thus, for this example, the system's recognition was errorless.

Pap-Smear Cells

We assembled a dataset of 23 normal and 16 abnormal Pap-smear cells (our Lahey Clinic collaborators judged the cell types). The images were 175 × 175 pixels, with 8-bit gray values. Figure 7(a) shows a typical image

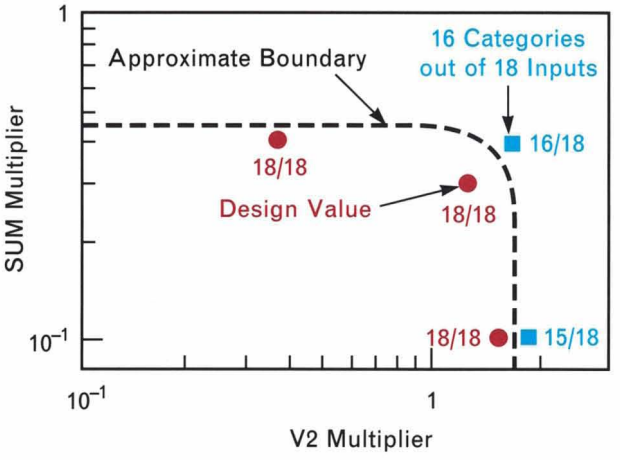


FIGURE 6. SUM and V2 multipliers (Figure 1) for an 18-image training set of military vehicles. Note that the number of categories was equal to the number of input patterns when we selected low values for both multipliers. As we increased the values of the multipliers, the system lost its ability to discriminate between certain categories, and the number of categories decreased below the number of inputs. The dotted line represents a rough boundary for this transition. We selected a design value inside the boundary and near the elbow of the curve.

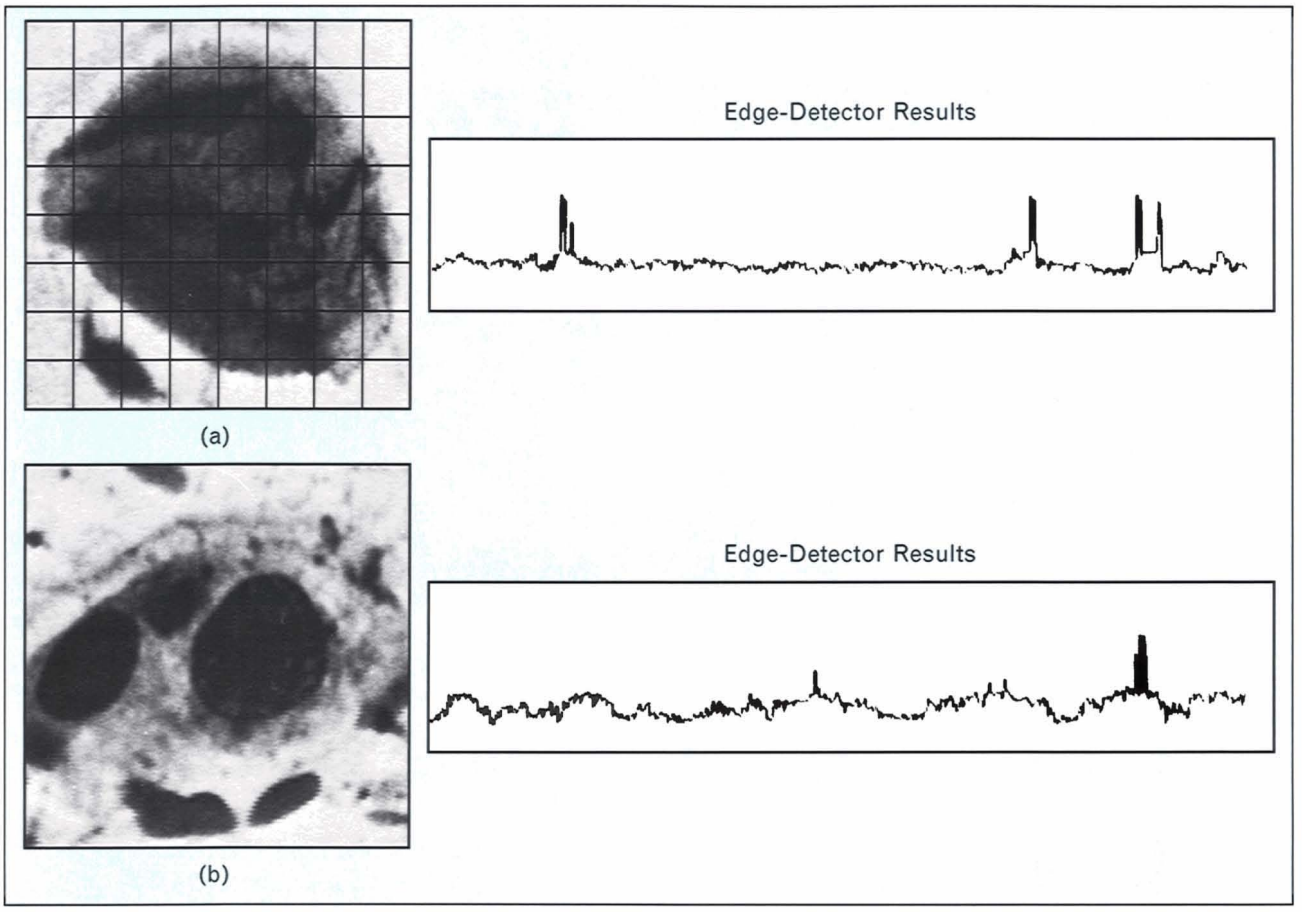


FIGURE 7. Comparison of (a) normal Pap-smear cell and (b) abnormal Pap-smear cell.

of a normal cell and Figure 7(b) a typical image of an abnormal cell (the grid suggests the V1 processing). To the right of the photographs, V1 feature values near the cell's nuclei are shown.

To train and test the system on different orientations, we rotated each cell image 90°, 180°, and 270°. The rotations expanded the dataset to 92 normal and 64 abnormal cells, or 156 altogether.

As with the military-vehicle example, we selected the multipliers to weight the SUM, V1, and V2 contributions about equally. We trained the system with a Vigilance of 0.99999, which corresponded to a 0.256° sepa-

Table 1. Preliminary Classification Results for Military Vehicles

	System Classification		
	Tank	*Howitzer*	*Unknown*
Vehicle			
Tank	10	0	0
Howitzer	0	8	0
Armored Personnel Carrier	0	0	4

Note: For training, we used 10 tanks and 8 howitzers.

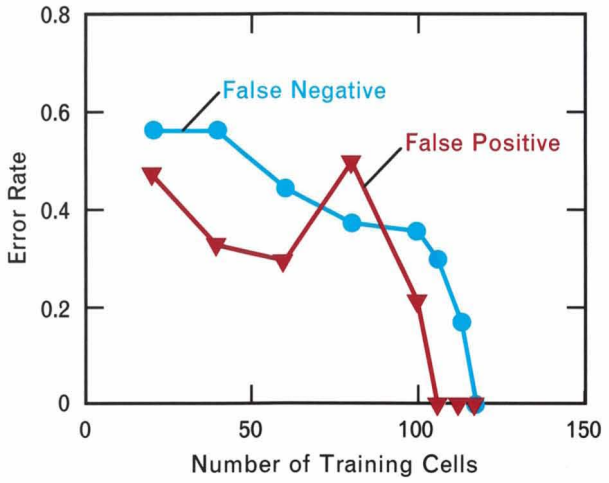


FIGURE 8. Error rate versus training-set size for images of Pap-smear cells.

ration in feature space. (Tests showed that the separation of the cells in feature space varied from 0.256° to 30°.)

Initially we chose the training sets randomly, as we had done with the military-vehicle dataset. The random selection, however, resulted in high error rates; i.e., some cells did not make good training examples. In general, those cells far from the normal-abnormal boundary in feature space did not help the system improve its discrimination ability. For this reason, we developed an iterative training method that selects cells near the boundary.

To train the system iteratively, we started with two normal and two abnormal cells. We then tested the system on the remaining 152 images. We increased the training set by adding roughly equal numbers of false positives and false negatives. (False positives are normal cells that have been classified as abnormal. False negatives are abnormal cells that have been classified as normal.) The error rates of false positives and false negatives dropped as we repeated the procedure.

Figure 8 shows the error rates as we varied the number of training examples. We should note that it is crucial to keep the false-negative rate small to avoid potentially fatal errors. The curves were produced by the iteration method described above.

The results of Figure 8 suggest the system generalizes from its training. Mathematically speaking, the feature vectors lie in a 2505-dimensional vector space, as described earlier in the subsection "Feature Vector." The normal cells lie in a subset of that vector space, and the abnormal cells lie in a different subset. If the subsets had formed a checkerboard pattern or had a highly jagged boundary, training might have required all the images. However, the curve in Figure 8 suggests that we can eliminate all false negatives with far fewer images. Thus we believe that the boundary is comparatively smooth, which allows the system to generalize. Table 2 summarizes our results; they show no false positives or false negatives with 118 training images.

The results suggest that the system might have promise for initial cytology screening. Furthermore, the results suggest that the error rate can be decreased to less than 5%, for example, with training sets of several hundred examples for each cell type. More testing is necessary both to confirm these preliminary results and to assess the system's practical value. For the system to achieve an error rate of less than a few percent, a much larger database is required.

Applications and Extensions

Reliable MV systems have many applications. Besides those areas of interest to MIT Lincoln Laboratory in

Table 2. Preliminary Cytology Results with Iterative Training

	System Classification	
	Normal	*Abnormal*
Cell Type		
Normal	26	0
Abnormal	0	12

Note: For training, we used 66 normal cells and 52 abnormal cells.

remote sensing and automatic target recognition, other uses include medical screening, industrial inspection, and robot vision. The architecture of our system is applicable to these diverse areas.

The basic system architecture is also extendable, and the following subsections describe several possible extensions. We should note that the examples include new principles and so are speculative.

Sensor Fusion

A direct extension of our research is to combine parallel sensors. Figure 9 shows a fusion concept at the feature-vector level. The bottom system is our basic system with minor additions, and the top is another system, which can be a different type.

The two systems produce features that train the classifier. Using video and laser-radar range images, we have done preliminary tests of this concept [6].

Moving Objects

Another extension is the capability to track and recognize moving objects. Figure 10 shows a conceptual block diagram in which an object in the FOV is moving in an arbitrary direction. To detect this motion, we can add modules that are sensitive to the motion of edges at multiple orientations. These motion detectors would mimic the characteristics of biological vision systems.

In Figure 10, the system feeds signals from the motion detectors back to the SUPERC module for tracking, and the motion-detection features are stored in the feature vector for recognition. To use time-varying features for recognition, we can replace the ART-2 module by an Avalanche neural network [7]. This modification would enable the recognition of, for example, a flying butterfly [8].

Binocular Vision

For an extension to binocular vision, Figure 11 shows a block diagram of two of our basic systems working in parallel. In the figure, the SUPERC module points the two "eyeballs," and the left and right FOV of each sensor (denoted as L1, L2, R1, and R2, respectively) go separately to two LGNs for calibration and normalization.

The system uses parallel sets of feature detectors, and the feature vector consists of the left and right features and their difference, or disparity. The clas-

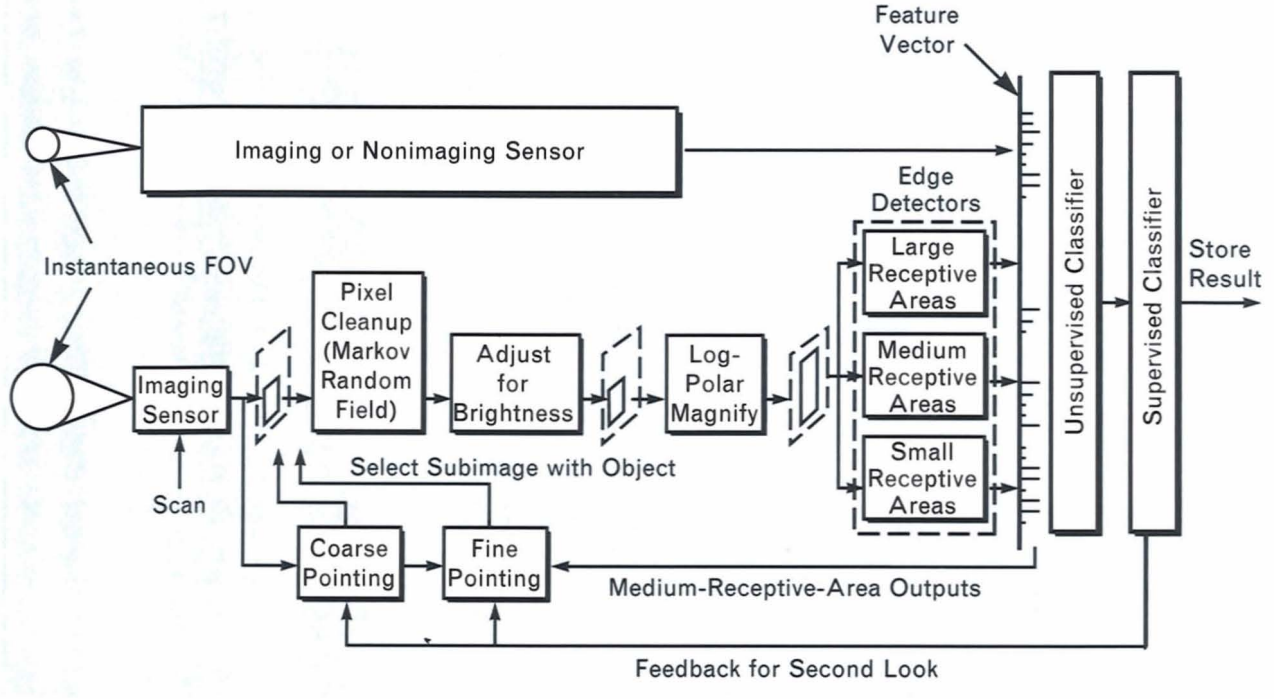


FIGURE 9. Block diagram showing a system that combines two parallel sensors.

sifier is similar to the classifier of the basic system.

Vision-Motor Systems

We end on a more speculative note by describing a neural network system for driving a car. Figure 12 shows the conceptual block diagram.

The video sensor, with two pointing angles, is part of the basic vision system. (We should note in passing that Boston driving requires more than one camera.) An addition to the vision system is the GAZE channel, which gives direction information that combines with the visual features to form the feature vector. The ITC is like that of the basic system: its output drives a command-generating neural network. The network used is Vector Integration to Endpoint (VITE) [9], a type of neural network that models biological motor systems. During the learning process, the adaptive elements are in the network module. Output from the command-generating neural network drives other modules that give speed, steering, and braking commands to the automobile.

Summary

We have developed a general-purpose machine vision (MV) system for recognizing stationary visual objects in their natural settings. The system uses neural networks and standard processing to model selected functions of human vision. The recognition is experiential, i.e., based solely on prior examples, and the system performance improves through experience.

We tested the system with images of military vehicles and human cervical smears, and the results were very encouraging. In fact, the results suggest that a practical system might be feasible.

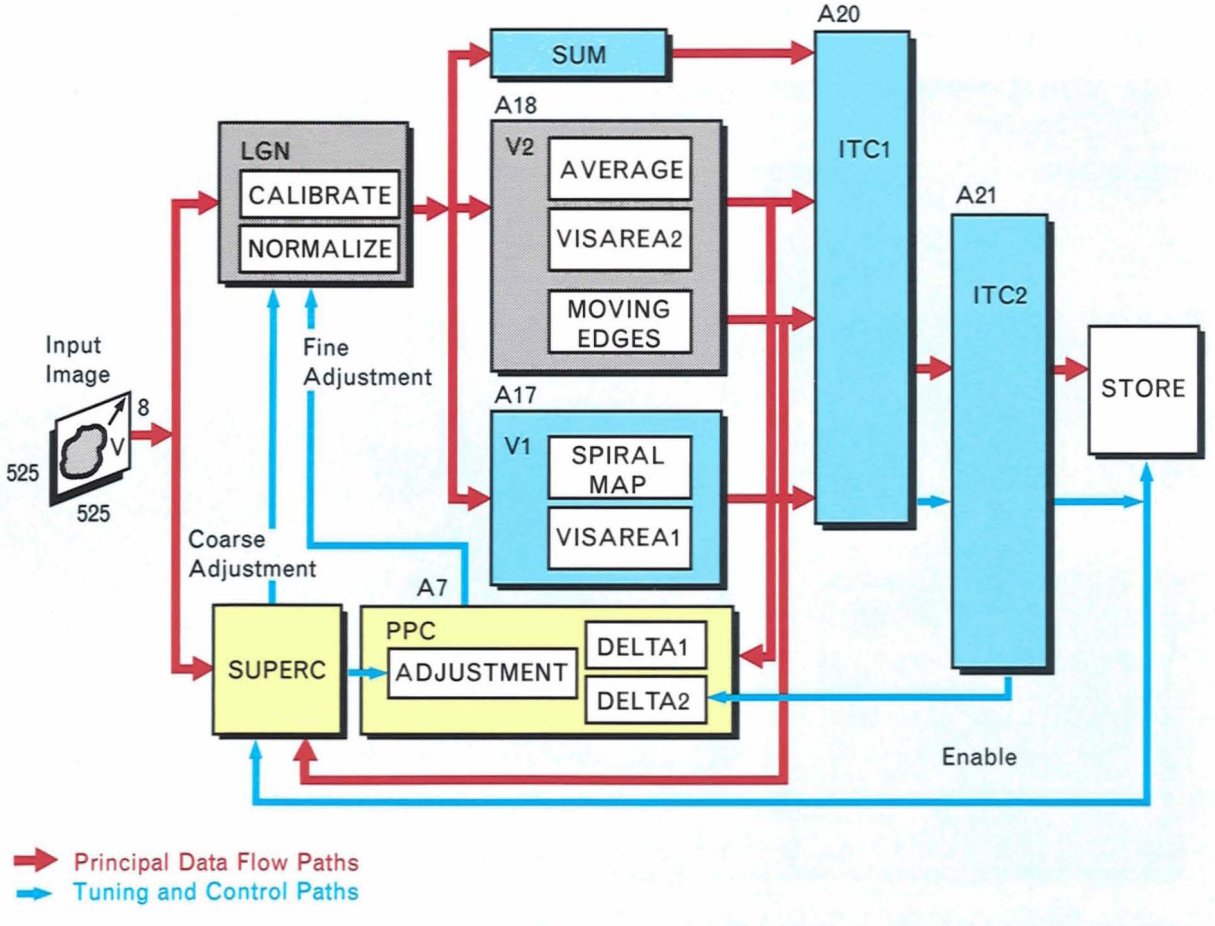


FIGURE 10. Block diagram of neural network architecture for moving objects. (Note: See the box "Glossary of Acronyms" for definitions of the acronyms used.)

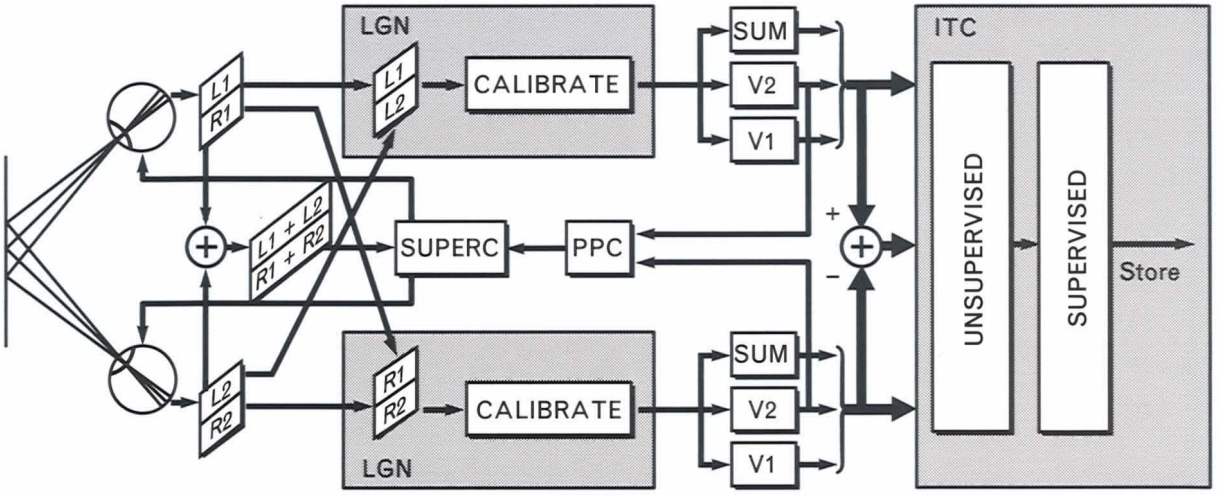


FIGURE 11. Block diagram of neural network architecture for binocular vision. (Note: See the box "Glossary of Acronyms" for definitions of the acronyms used.)

We believe this approach to MV is promising for many applications. At Lincoln Laboratory we are studying improvements that include motion detection, application to microwave radar and passive/active infrared imagery, and integration into complex systems. We are also considering the hardware implementation of selected modules.

Acknowledgments

The Innovative Research Program at Lincoln Laboratory supported this work from February 1989 to January 1990. Our supervisor, Robert Rafuse, gave us much encouragement.

We wish to acknowledge our Lincoln Laboratory colleagues Mary Fouser, Al Gschwendtner, Pat Hirschler-Marchand, Paul Kolodzy, Gloria Liias, Courosh Mehanian, Murali Menon, and Alex Sonnenschein for their help and valuable discussions.

We are also grateful to Prof. John Uhran, Jr., of the University of Notre Dame, and Dr. Barney Reiffen and Prof. Mike Carter of the University of New Hampshire for their consultation during this work. Special thanks go to our Lahey Clinic Medical Center collaborators Drs. Mark Silverman and John Dugan.

This work was supported by the Department of the Air Force.

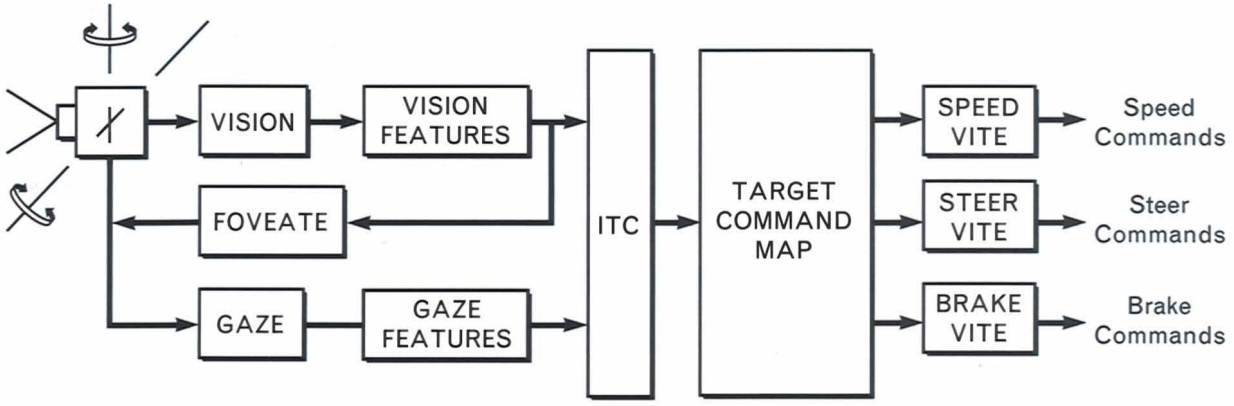
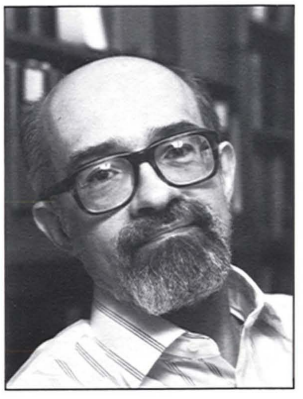


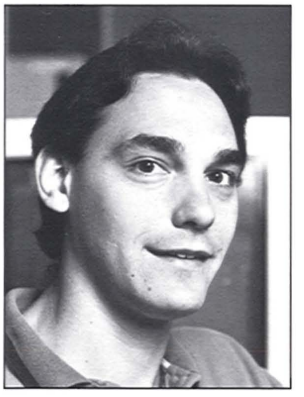
FIGURE 12. Block diagram of neural network architecture for driving a car. (Note: VITE, or Vector Integration to Endpoint, is a type of neural network that models biological motor systems.)

REFERENCES

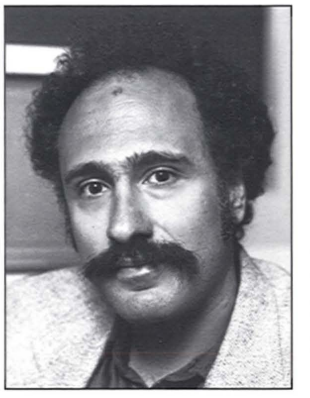
1. A. Rosenfeld, "Computer Vision: Basic Principles," *Proc. IEEE* **76,** 863 (1988).
2. D.C. Van Essen and J.H.R. Maunsell, "Hierarchical Organization and Functional Streams in the Visual Cortex," *Trends Neurosci.* **6,** 370 (1983).
3. F. Crick and C. Asanuma, "Certain Aspects of the Anatomy and Physiology of the Cerebral Cortex," in *Parallel Distributed Processing, Vol. 2,* eds. J. McClelland and D. Rumelhart (MIT Press, Cambridge, MA, 1986), pp. 333–371.
4. R.L. Harvey, P.N. DiCaprio, and K.G. Heinemann, "A Neural Architecture for Visual Recognition of General Objects by Machines," *Technical Report,* MIT Lincoln Laboratory (to be published).
5. G.A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Appl. Opt.* **26,** 4919 (1987).
6. R.L. Harvey and K.G. Heinemann, "A Biological Vision Model for Sensor Fusion," *IEEE 4th Natl. Symp. on Sensor Fusion* (to be published).
7. S. Grossberg, *Studies of Mind and Brain* (D. Reidel, Boston, 1982).
8. K.A.C. Martin and V.H. Perry, "On Seeing a Butterfly: The Physiology of Vision," *Sci. Prog., Oxf.* **72,** 259 (1988).
9. D. Bullock and S. Grossberg, "Neural Dynamics of Planned Arm Movements: Emergent Invariants and Speed-Accuracy Properties during Trajectory Formation," in *Neural Networks and Natural Intelligence,* ed. S. Grossberg (MIT Press, Cambridge, MA, 1988), pp. 553–622.



ROBERT L. HARVEY
is a staff member in the Opto-Radar Systems Group, where his focus in research has been in space technology, sensor systems, and neural networks. Before joining Lincoln Laboratory in 1972, Bob worked for the Conductron Corp. He received the following degrees from the University of Michigan: a B.S.E. in aerospace and mechanical engineering, and a Ph.D. in aerospace engineering. He is a member of Tau Beta Pi, IEEE, the American Institute of Aeronautics and Astronautics (AIAA), and the International Neural Network Society (INNS).



PAUL N. DICAPRIO
received a B.S. degree in computer science from John Carroll University and an M.S. degree in computer engineering and science from Case Western Reserve University. Paul worked for Conley, Canitano and Associates Inc. before joining Lincoln Laboratory three years ago. An associate staff member in the Opto-Radar Systems Group, he specializes in connectionist models and interactive 3-D visualization.



KARL G. HEINEMANN
is an assistant staff member in the Opto-Radar Systems Group, where he designs algorithms and software for advanced sensor systems. He specializes in vision and neural networks. Karl worked at the Smithsonian Astrophysical Observatory before joining Lincoln Laboratory in 1979. He received a B.A. degree in physics from Swarthmore College and has done graduate work in astronomy at Cornell University. He is a member of the American Association for the Advancement of Science (AAAS) and the International Neural Network Society (INNS).