Pergamon

Neural Networks, Vol. 8, No. 7/8, pp. 1053–1080, 1995

0893-6080/95 $9.50+.00

0893-6080(95)00053-4

1995 SPECIAL ISSUE

Fast-Learning VIEWNET Architectures for Recognizing Three-dimensional Objects from Multiple Two-dimensional Views

GARY BRADSKI AND STEPHEN GROSSBERG

Boston University

(*Received and accepted* 3 *February* 1995)

Abstract—*The recognition of three-dimensional (3-D) objects from sequences of their two-dimensional (2-D) views is modeled by a family of self-organizing neural architectures, called VIEWNET, that use View Information Encoded With NETworks. VIEWNET incorporates a preprocessor that generates a compressed but 2-D invariant representation of an image, a supervised incremental learning system that classifies the preprocessed representations into 2-D view categories whose outputs are combined into 3-D invariant object categories, and a working memory that makes a 3-D object prediction by accumulating evidence from 3-D object category nodes as multiple 2-D views are experienced. The simplest VIEWNET achieves high recognition scores without the need to explicitly code the temporal order of 2-D views in working memory. Working memories are also discussed that save memory resources by implicitly coding temporal order in terms of the relative activity of 2-D view category nodes, rather than as explicit 2-D view transitions. Variants of the VIEWNET architecture may be used for scene understanding by using a preprocessor and classifier that can determine both* what *objects are in a scene and* where *they are located. The present VIEWNET preprocessor includes the CORT-X 2 filter, which discounts the illuminant, regularizes and completes figural boundaries, and suppresses image noise. This boundary segmentation is rendered invariant under 2-D translation, rotation, and dilation by use of a log-polar transform. The invariant spectra undergo Gaussian coarse coding to further reduce noise and 3-D foreshortening effects, and to increase generalization. These compressed codes are input into the classifier, a supervised learning system based on the fuzzy ARTMAP algorithm. Fuzzy ARTMAP learns 2-D view categories that are invariant under 2-D image translation, rotation, and dilation as well as 3-D image transformations that do not cause a predictive error. Evidence from sequences of 2-D view categories converges at 3-D object nodes that generate a response invariant under changes of 2-D view. These 3-D object nodes input to a working memory that accumulates evidence over time to improve object recognition. In the simplest working memory, each occurrence (nonoccurrence) of a 2-D view category increases (decreases) the corresponding node's activity in working memory. The maximally active node is used to predict the 3-D object. Recognition is studied with noisy and clean images using slow and fast learning. Slow learning at the fuzzy ARTMAP map field is adapted to learn the conditional probability of the 3-D object given the selected 2-D view category. VIEWNET is demonstrated on an MIT Lincoln Laboratory database of 128×128 2-D views of aircraft with and without additive noise. A recognition rate of up to 90% is achieved with one 2-D view and of up to 98.5% correct with three 2-D views. The properties of 2-D view and 3-D object category nodes are compared with those of cells in monkey inferotemporal cortex.*

Keywords—Pattern recognition, Learning, Neural network, ARTMAP, Inferotemporal cortex.

1. VIEW INFORMATION ENCODED WITH NETWORKS

This article describes a neural architecture, called VIEWNET, for autonomously learning to recognize three-dimensional (3-D) objects in real time. More generally, VIEWNET illustrates a computational strategy that may be generalized to define architectures capable of more complex task of scene understanding. VIEWNET derives its name from its ability to use View Information Encoded With

Acknowledgements: The authors wish to thank Diana Meyers for her valuable assistance in the preparation of the manuscript.

GB was supported in part by the National Science Foundation (NSF IRI-90-24877) and the Office of Naval Research (ONR N00014-92-J-1309), and SG was supported in part by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0499) and ARPA (AFOSR 90-0083 and ONR N00014-92-J-4015).

Requests for reprints should be sent to Professor Stephen Grossberg, Boston University, Center for Adaptive Systems, 677 Beacon Street, Boston, MA 02215, USA.

NETworks. This computational strategy attempts to achieve recognition of a 3-D object by learning invariant properties of its two-dimensional (2-D) views and then accumulating evidence from sequences of 2-D views until 3-D recognition is assured. Such an approach uses only the types of information that are available to the system during its ongoing encounters with objects. The ultimate goal is to define an autonomous system for general-purpose object recognition and scene understanding.

Many prior approaches to 3-D visual object recognition sought to build up a structural description of an object to aid in the recognition process. Quite a few of these efforts used volumetric or surface-based descriptions to generate structural models of 3-D objects. Winston (1975) used multiple 2-D views to create structural models of objects in a scene. He used the spatial relationships between simple primitives such as "wedge supported by brick" in each view to construct his models. Subsequent views were used to reinforce or modify the model. Underwood and Coates (1975) incrementally built a surface description of an object from multiple views for recognizing polyhedra. Freeman and Chakravarty (1980) used characteristic views in distinct topological arrangement as an index for identifying 3-D objects. Martin and Aggarwal (1983) built up a volume segment model of 3-D objects from 2-D views. Using explicit information about each view point, they were able to use learning to refine the descriptions of the 3-D objects. Fekete and Davis (1984) tessellated a sphere and used the resulting viewpoints to index the first and second moment properties of images. No consideration was given to learning however, and no use was made of tessellation sequences for recognition.

Chen and Freeman (1990) extended the method of Freeman and Chakravarty by using volumetric descriptions of 3-D objects to predict topologically distinct characteristic views of quadric-surfaced solid models. Edelman et al. (1989) modeled human performance in mental rotation experiments using a neural network and 2-D views of 3-D wire frame objects. Liu and Tsai (1990) learned a sequence of 2-D silhouettes from rotating 3-D objects as a series of moments which could then be matched against a test series for recognition. Training and testing data sets were derived by rotating objects on a turntable so that arbitrary object rotations were not handled. Rather, they intended their technique to be employed in a controlled viewing environment such as could be set up in a factory. Zhang et al. (1992) also constructed a 3-D object model from 2-D view relations. In an attempt to manage the combinatorial complexity of using 2-D aspects, Dickinson et al. (1992) use a fixed set of 3-D volumetric primitives to describe 3-D objects. The 3-D volumetric primitives are used to generate a hierarchy of 2-D view aspects to which 2-D images are matched. The 2-D aspect hierarchy was intended to handle 3-D occlusions so that an input 2-D image indexes the set of 3-D primitives which in turn indexes the 3-D object being viewed.

A number of researchers, concerned about the recognition speed limitations of volumetric or surface model matching methods, have concentrated on appearance-based approaches. Appearance-based recognition approaches use input imagery to construct 3-D object models. Perhaps the earliest explicit use of imagery to construct structural representations of 3-D objects was developed by Koenderink and van Doorn (1979). Koenderink and van Doorn created *Aspect Graphs* consisting of 2-D views of a 3-D object along the nodes of the graph, with legal view transitions indicated by the arcs among nodes as shown in Figure 1. In the figure, views of a cube (say a numbered dice) are displayed with arcs representing their topological relationship to one another. For Koenderink and van Doorn, 2-D views and view transitions were equally important for recognizing the object.

In the past few years, with inexpensive computer power and memory increasingly available, there has been growing interest in methods of automatically generating representations from imagery akin to Koenderink and van Doorn's aspect graphs. Gigus and Malik (1988, 1990), and Plantinga and Dyer (1990) have attempted to automatically construct aspect graphs from objects in a CAD database using convex polyhedra. Other efforts for automatically generating perspective projection aspect graphs from a CAD database using curved objects and non-convex polyhedra have been pursued by Bowyer et al. (1989), Sripradisvarakul and Jain (1989), Ponce and Kriegman (1990), Rieger (1990), Stewman and

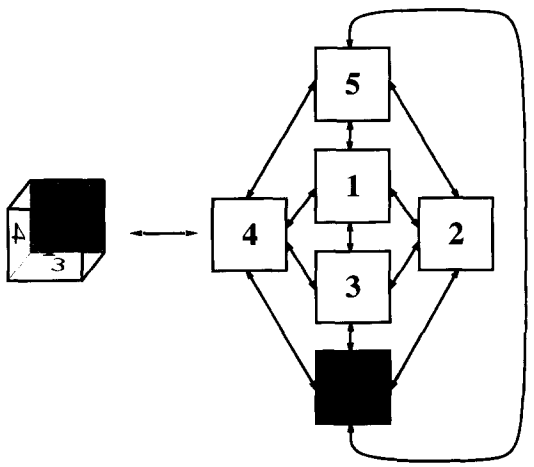

FIGURE 1. Aspect Graph of Koenderink and van Doorn. Each node of the graph represents a distinct 2-D view of the object. Arcs represent legal view transitions that would be observed when viewing the object.

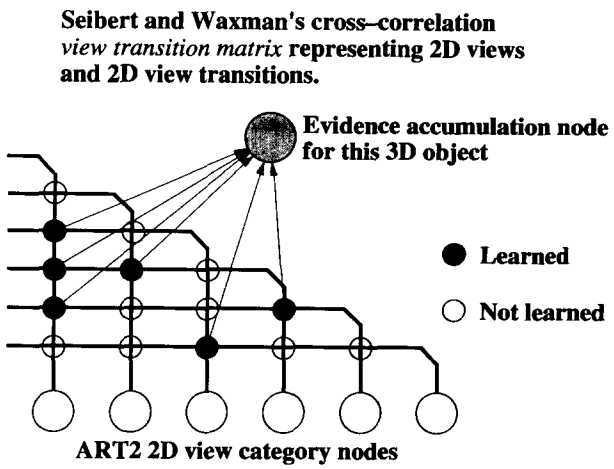

FIGURE 2. View transition matrix from the architecture of Seibert and Waxman computes cross-correlations between its input 2-D view categories. The matrix represents the correlation between the present categorical 2-D view and the decaying activation of the previous view. As preprocessed images enter ART 2, the resulting categorical views excite learned weights in this cross-correlation "aspect graph" devoted to each 3-D object. An evidence node integrates activation from the learned connections in the matrix. Another network (not shown) chooses the evidence node with the highest accumulated activation as the 3-D object most likely being viewed by the architecture.

Bowyer (1990), Chang and Huang (1992), and Eggert and Bowyer (1993). Hidden Markov models have also been applied to learning an aspect graph from a view sequence by Rimey and Brown (1991).

Seibert and Waxman (1990a, b, 1991, 1992) pioneered the development of neural network architectures that self-organize aspect graph representations of 3-D objects from 2-D view sequences. Since our work on VIEWNET was inspired by their model and uses the same database to benchmark our results, we review their model in some detail. Their approach relied on building a neural "cross-correlation matrix" as shown in Figure 2. The cross-correlation matrix was used to learn both 2-D views and 2-D view transitions and to associate the 2-D views and view transitions with the 3-D objects that produced them. To test their architecture, the 2-D views were taken from 3-D jet airplane models painted black and digitized against a light background for ease of segmentation.

Sequences of images were obtained by rotating the jets 360 degrees for each of the many angles of inclination that the video camera was set at, from zero degrees (horizontal) to 72 degrees (looking down at the spinning jet). These images were then binarized using a high threshold to remove noise. Points of high curvature and the object centroid were found using a reaction-diffusion process. A log-polar transform around the object centroid was used to remove 2-D rotation and scale variations. The output of this filter was (approximately) invariant under 2-D translations, rotations, and dilations of the image. In order to compress this invariant spectrum and reduce 3-D foreshortening effects, the result was coarse coded (compressed to 5×5 pixels from 128×128) using Gaussian filters.

The coarse-coded vectors (25 data points) were fed into an ART 2 (Carpenter and Grossberg, 1987) network for unsupervised learning and categorization. These "categorical" 2-D views further compressed the 2-D representation so that a new 2-D view category was chosen only if significant changes occurred in the 2-D appearance of the object that were not invariant under translation, rotation, dilation, or modest foreshortening. How much change was tolerated was controlled by the ART 2 vigilance parameter. These 2-D view categories were then fed into a series of cross-correlation matrices, or view graphs, one for each possible 3-D object, so that views and view transitions could be learned by a 3-D object categorization layer. The 3-D categorization layer incorporated "evidence accumulation" nodes which integrate activations that they receive from learned connections to the correlation matrix. Decay terms in these integrator nodes determine how long they stay active without input support.

Seibert and Waxman's approach of automatically generating aspect graphs directly from the imagery that the architecture experiences vastly simplifies earlier attempts which generated aspect graphs by constructing projections from mathematical descriptions of the 3-D objects. However, using view transition information comes at a cost: Given N 2-D views and M objects, the architecture must have the potential to encode an order of $N^2 \times M$ 2-D view transitions and the corresponding adaptive weights. Another potential problem is that an error in identifying a 2-D view may introduce a spurious 2-D view transition. Finally, unless one presumes a 2-D view frame capture rate fast enough to capture the highest speed movement that an object can make, view transitions may be skipped inadvertently by fast object motion.

As reported in Seibert and Waxman (1992), 75% of the 2-D jet images were ambiguous to some degree. That is, 75% of the 2-D view categories formed by ART 2 gave evidence for more than one type of jet. Two possible reasons for this level of ambiguity exist: (1) image preprocessing using high curvature points followed by coarse coding may lump together object features that are needed for unambiguous recognition; and (2) the 2-D views were categorized by ART 2 without using any supervised feedback to help correct category boundaries. Although most 2-D view categories were ambiguous, the transitions between them were used to unambiguously identify a particular 3-D object. Thus, view transitions are critically important in the Seibert and Waxman architecture, which may then incur the cost of

needing up to $N^2 \times M$ view transition correlation matrices.

This analysis suggests that a tradeoff exists between the choice of preprocessor, learned categorizer, and evidence accumulation parts of the architecture. If the preprocessor and categorizer generate 2-D view categories that are too coarse or ambiguous, then the evidence accumulation network may have to be enhanced to overcome these limitations. VIEWNET explores this tradeoff by using a different preprocessor and categorizer that generate less ambiguous 2-D view categories. In fact, VIEWNET can categorize individual views on the Seibert–Waxman database with high accuracy (up to 90%), in accord with the human experience that many objects can be identified with a single view, except when they are observed from an unfamiliar perspective or from a perspective that reduces the object's apparent dimension. A computationally less costly evidence accumulation, or working memory, network could then be used, at least on these data. The general problem is to design the optimally balanced preprocessor, categorizer, and working memory networks to handle the largest possible set of images.

The VIEWNET approach was also motivated by work of Bradski et al. (1992), who described an alternative architecture that could potentially overcome the problem of proliferating view transitions. This architecture learns to code 2-D views in recognition categories, as do Seibert and Waxman, but stores these categories in a working memory, called a STORE network, whose activity pattern codes N view categories and their temporal order using only N codes. Larger activation of a category node in a STORE working memory codes for earlier occurrence of the corresponding 2-D view. An ART network then learns to categorize the stored combination of 2-D views and (implicitly coded) view transitions into a 3-D object category. Such an algorithm needs no more than $N \times M$ adaptive weights to code N 2-D views in working memory for M 3-D objects. This paper further develops the perspective that, although multiple views may facilitate recognition, explicit coding of view transitions may not be needed to achieve high recognition accuracy.

As diagramed in Figure 3, VIEWNET consists of three parts: an image preprocessor, a self-organizing recognition network that may operate in either unsupervised or supervised modes, and a working memory network to accumulate evidence over multiple views. It is assumed that the figure to be recognized has already been separated from its background. Neural networks for figure-ground separation that use computations consistent with those in the VIEWNET preprocessor were described

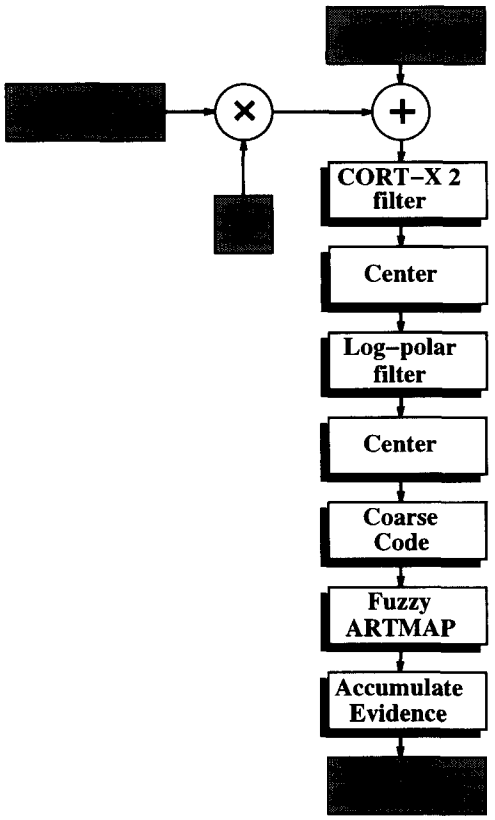

FIGURE 3. The image processing flow chart of the VIEWNET system.

in Grossberg (1994) and Grossberg and Wyse (1991, 1992). The image figure is then processed by a boundary segmentation network, called the CORT-X 2 filter (Carpenter et al. 1989; Grossberg and Wyse, 1991, 1992). CORT-X 2 is a feedforward network that first compensates for variable illumination, extracts ratio contrasts, and normalizes image intensities. It then suppresses image noise while it completes and regularizes a boundary segmentation of the figure. Thus the maximal curvature point representation of Seibert and Waxman is replaced by an illumination-compensated, noise-suppressed segmentation of the entire figure. This boundary segmentation is then rendered invariant under 2-D rotation, translation, and scale by a centering, log-polar, centering operation as described in Schwartz (1977). As in Seibert and Waxman, the resulting spectra are then compressed by coarse coding to gain some insensitivity to 3-D deformation effects and to reduce memory requirements. Coarse coding is done by two methods, whose performance is compared: a many-to-few pixel simple spatial averaging, and Gaussian spatial averaging. The output of this preprocessor is a coarse-coded, invariant spectrum of an illumination-compensated, noise-suppressed boundary segmentation. This representation provides the input vectors to the self-organizing neural network classifier.

Fuzzy ARTMAP (Carpenter et al., 1992a) was used to categorize the output spectra. This architecture is capable of fast, stable learning of recognition categories in response to nonstationary multidimensional data, and of learning to generate many-to-one output predictions from the recognition categories to output labels. Fuzzy ARTMAP runs under either unsupervised or supervised learning conditions. Under supervised conditions, erroneous predictions trigger further hypothesis testing, or memory search, in the input classifier. Fuzzy ARTMAP converts the vigilance parameter of unsupervised ART classifiers, such as ART 2, into an internally controlled parameter. When an erroneous prediction occurs, vigilance is increased just enough to trigger a new bout of hypothesis testing to discover a better category. This control scheme is called *match tracking* because the vigilance parameter tracks the match value that encodes how well the selected category's prototype matches the input spectrum. Using match tracking, memory search discovers and learns recognition categories that conjointly maximize code compression and minimize predictive error. Fuzzy ARTMAP can hereby use supervised learning to rapidly fit the number, size, and shape of input categories to the statistical demands of the environment. This added power helps Fuzzy ARTMAP to learn 2-D view categories that tend to fit the data better than ART 2. In the simplest VIEWNET, that we call VIEWNET 1, Fuzzy ARTMAP also automatically combines the outputs of 2-D view categories at 3-D object category nodes that are invariant under changes of experienced 2-D views. It is this combination of boundary segmentation preprocessing combined with supervised ART learning that achieves correct prediction of up to 90% accuracy in response to a single airplane view.

A similar type of hierarchical organization from 2-D view to 3-D object has been reported in neurophysiological studies of cell responses in monkey inferotemporal cortex, where some cells respond to individual 2-D views whereas others, like 3-D object nodes, respond to a wide range of views (Logothetis et al., 1994). These studies were motivated by the regularization networks of Poggio and Edelman (1990) which also add up responses from 2-D views at 3-D object nodes. These networks do not, however, incrementally learn their categories in real-time and have not yet been incorporated into a self-organizing architecture.

Given the high accuracy attained by individual 2-D view categories, the simplest possible working memory was used in VIEWNET 1 to illustrate the tradeoff between preprocessor, categorizer, and working memory. No view transitions were used. In fact, no temporal order information was used. Instead, the working memory simply updated its representation of each 3-D object category every time one of its 2-D views was experienced, and the 3-D object was predicted by voting for the winning 3-D category. Using this scheme, voting with two views achieves up to 94%, and with three views up to 98.5% accuracy. It was hereby shown that the simplest evidence accumulation from multiple views, without view transitions in a VIEWNET 2 architecture or even a representation of view temporal orders, can lead to high recognition on the Seibert–Waxman database if the preprocessor and classifier are designed as indicated.

The remainder of the paper describes these operations in detail. Section 2 describes the airplane database. Section 3 describes the CORT-X 2 boundary segmentation network and Appendix A describes its equations. Section 4 describes the operations to generate an invariant representation of the boundary segmentation. Section 5 describes the coarse-coding algorithm. Section 6 describes the Fuzzy ARTMAP network and Appendix B describes the equations. Section 7 describes computer simulation results in response to a single view, using fast or slow learning with or without image noise. The results are robust across all simulation conditions. Section 8 summarizes how multiple views may improve recognition scores, and demonstrates that learning explicit view transitions in a VIEWNET 2 architecture does not improve performance on this database. Section 9 discusses these results in a broader context and outlines how these computations may be generalized to define an architecture for scene understanding.

2. DATA

The image database used to test the architecture described below consists of multiple 2-D images of three airplanes: an F-16, F-18, and HK-1 (we thank Michael Seibert and Allen Waxman of MIT Lincoln Laboratory for the use of their data and their unfailing cooperation). Video images were taken of three models of these airplanes. Each was painted black and suspended by string against a light background to aid in segmentation. The camera was mounted anywhere in an arc around the jets that started at 0.0 degrees above horizontal and went in increments of 4.5 degrees to a maximum of 72.0 degrees above horizontal. For each camera angle, the jets were spun and frames covering one full revolution (an average of 88 frames) were retained resulting in 1200 to 1400 images per object. The images themselves were 128×128 pixel gray-scale. The images were then thresholded and binarized into a SUN raster format to form the "raw" database. For our processing, data was turned into a floating point

format scaled between 0.0 and 1.0 and an additive noise process was introduced. The noise consisted of a 128×128 pixel image with each pixel taken from a uniform distribution between 0.0 and 1.0 scaled by a constant $C \geqslant 0.0$. That is, every pixel in the noise image is multiplied by C to create a "scaled" noise image. Thus, if say $C = 0.5$, the noise image would consist of pixels that varied randomly in amplitude with uniform distribution between 0.0 and 0.5. These scaled, 128×128 noise images were then added to the 128×128 airplane images prior to processing. Thus, both noise-free and noisy 2-D views covering a half-sphere surrounding the 3-D object were collected, keeping their spatial relationships intact.

Even-numbered rotation images from each camera angle were taken as the training set with the odd-numbered images forming the test set. The system was trained using random walks over the half-sphere of training images. Testing was done using random walks over the half-sphere of test images so that the paths taken and views seen were never the same between the training and test sets. Figure 4 shows example image from the database. The difficulty of the problem derives in part from the existence of small ambiguous frontal views in the database; see Figure 13 which appears later in this paper.

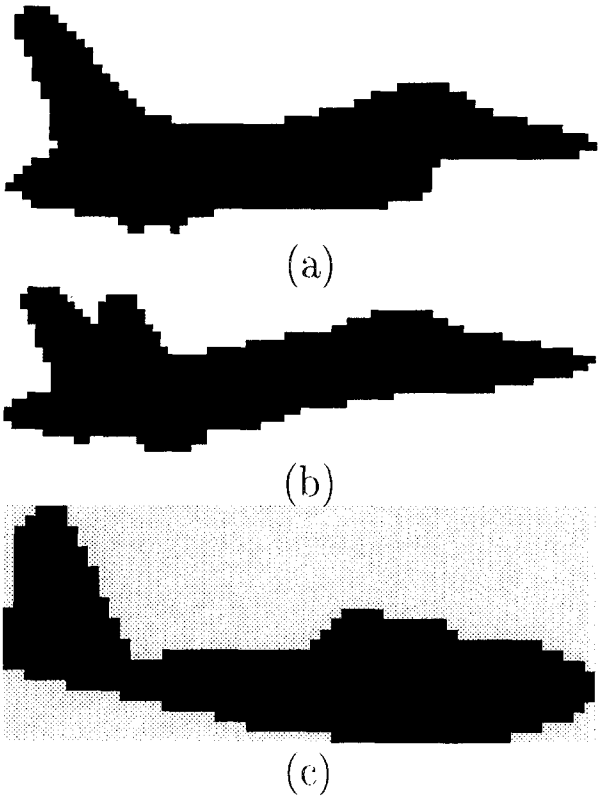

FIGURE 4. Example images from the database. Side views of (a) an F-16, (b) an F-18, and (c) an HK-1.

3. CORT-X 2 FILTER

The CORT-X 2 filter was used to preprocess the 128×128 airplane images. This is a feedforward network that detects, regularizes, and completes image boundaries from edge, texture and shading contrasts, while suppressing noise that does not have an underlying anisotropic structure. The CORT-X 2 filter is an enhancement of the original CORT-X filter (Carpenter et al., 1989). It generates better boundary segmentations, deals better with noise, and may also be used for figure-ground separation. The figure-ground separation properties were not needed in this research because the data images were already separated from their backgrounds. These CORT-X filters are simplifications of the Boundary Contour System (or BCS) for boundary segmentation (Grossberg, 1987, 1994; Grossberg and Mingolla, 1985; Grossberg et al., 1989, 1995). The BCS includes internal feedback loops to generate coherent boundary completions even over image regions defined by sparse image contrasts. A full BCS system can be inserted into the VIEWNET architecture to handle a wider class of imagery.

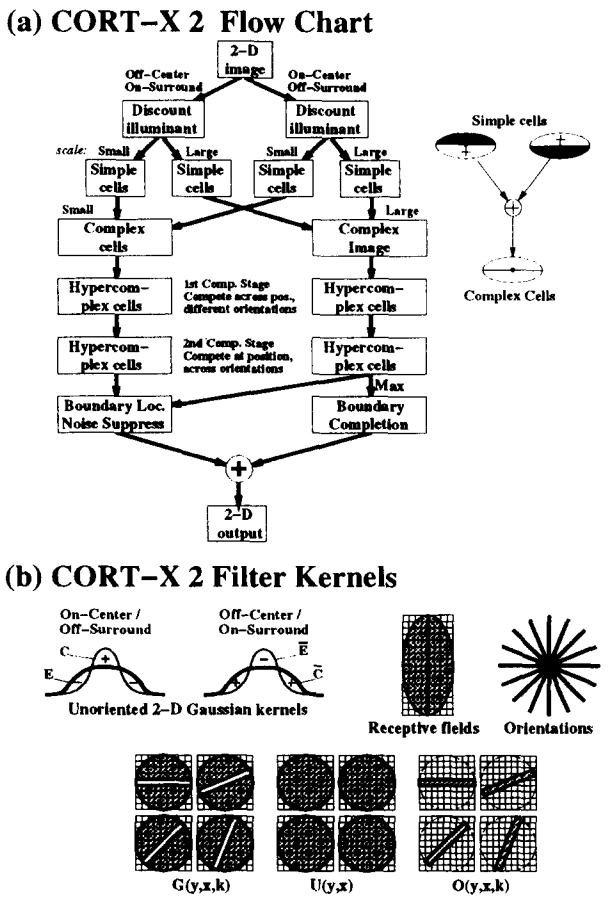

FIGURE 5. CORT-X 2 flow-chart and filter kernels. The image is processed in parallel with small and large scale filters. Grey areas in the kernels are the active regions. All kernels are normalized to have an area equal to one.

The CORT-X 2 filter incorporates a number of features that are useful for processing noisy imagery, including ON and OFF cells and multiple size scales, that respond to images in complementary ways. The CORT-X 2 filter embodies a computational strategy for combining these complementary computations to achieve enhanced image processing. The processing stages of the CORT-X 2 filter are schematized in Figure 5 and summarized below. Equations and parameters are listed in Appendix A.

Step 1. Discount the Illuminant

The first processing stage compensates for variable illumination, and thereby extracts ratio contrasts from the image while normalizing the overall activity level in each image region. Both ON cells and OFF cells process the image in this way, using parallel shunting on-center/off-surround ("ON-C") and off-center/on-surround ("OFF-C") networks. Parameters are set so that the ON-C network has a zero baseline activity and the OFF-C network has a positive baseline activity. The OFF-C filter performs an image inversion because it has a positive baseline activity that is inhibited by positive signal values in the image. Figure 6 shows a noise-free image as well as the ON-C and OFF-C outputs.

Along straight contrast boundaries in an image, both the ON-C and OFF-C networks enhance the contrast. On the other hand, the ON-C network has a stronger response to concave corners of activity in an image than the OFF-C network, while the converse is true at convex corners, as was noted by Grossberg and Todorović (1988). These complementary responses are joined at later processing stages used to build more complete boundary segmentations.

Step 2. Boundary Segmentation

The CORT-X 2 filter transforms the normalized ON-C and OFF-C images into a boundary segmentation using fast feedforward computations with oriented contrast-sensitive cells. Each processing stage possesses a full range of oriented cells as at each pixel in the image. Image processing is done in parallel using two sets of convolution kernels at two different size scales. As noted in Carpenter et al. (1989), larger scale oriented cells are better able to complete gaps in image boundaries and to suppress noise. On the other hand, smaller scale oriented cells do a better job of

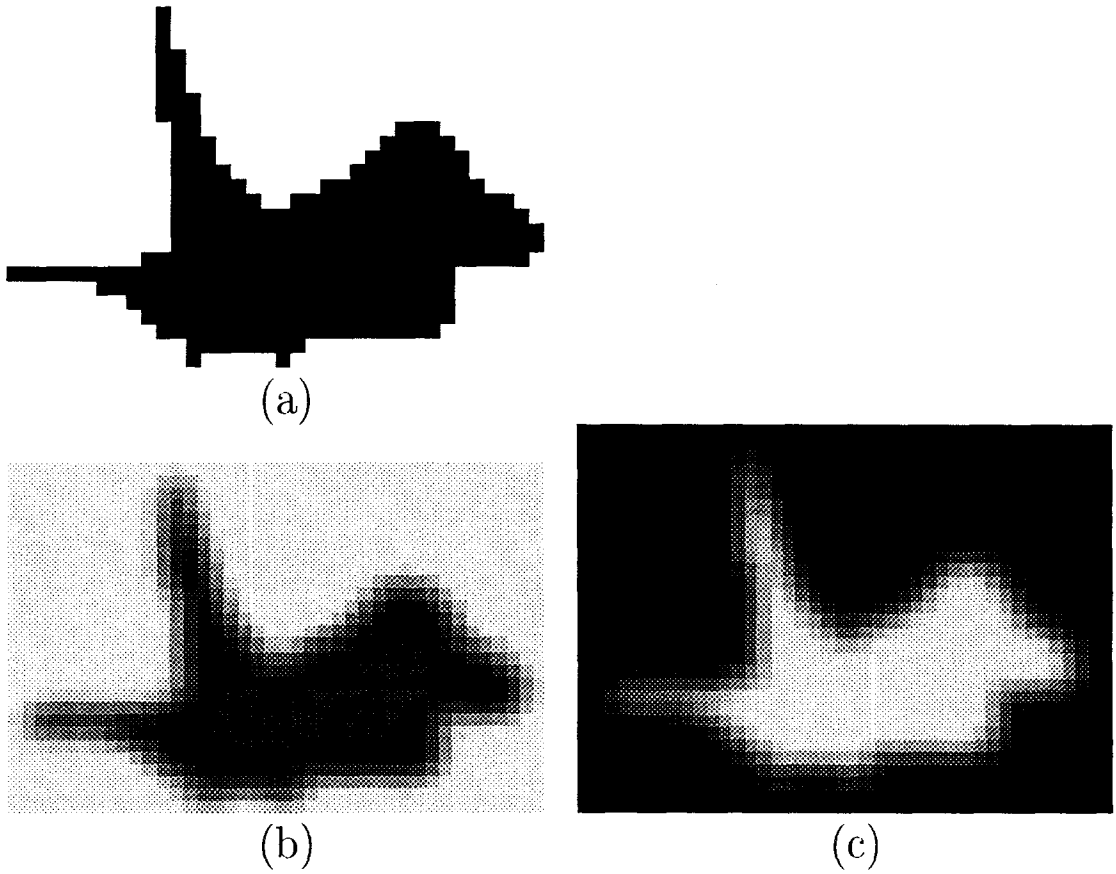

FIGURE 6. (a) The original F16 image. (b) CORT-X 2 ON-C output. (c) CORT-X 2 OFF-C output.

boundary localization than do larger scale cells. Interactions between the two size scales are designed to generate boundary outputs that combine good localization, boundary completion, and noise suppression properties.

The first stage, called the *simple cell layer*, consists of oriented contrast detectors that are sensitive to the orientation, amount, direction, and spatial scale of image contrast at a given image location. The orientation sensitivity results from using an elliptically shaped kernel, or input field, one for each of eight orientations spaced 45 degrees apart that operate in parallel at each position in the image. Sensitivity to direction-of-contrast results from a kernel in which one half is excitatory and the other half inhibitory. At each orientation, a pair of detectors sensitive to opposite directions-of-contrast processes the image. The net activity of each detector is rectified, giving rise to a half-wave rectified output signal. Figure 7 shows the results of processing the ON-C (Figure 7a) and OFF-C (Figure 7b) image with the small spatial scale (6 × 3 pixels) simple cell layer. The line lengths in the figure indicate the magnitude of the simple cell responses at each orientation and position.

The next processing stage generates a cell type whose output is insensitive to direction-of-contrast, or contrast polarity. Such an operation enables image boundaries to bridge textured and shaded image regions where contrast polarity reverses. This *complex cell layer* combines outputs from the simple cells at each position as shown in Figure 5. Complex cells perform a full-wave rectification of the image that is sensitive to the orientation, amount, and spatial scale of the contrast in the image, but not to its direction-of-contrast. To achieve this, complex cells sum up the half-wave rectified outputs of like-oriented and scaled simple cells of both directions-of-contrast at each position from both the ON-C and OFF-C networks. This is done in two parallel circuits

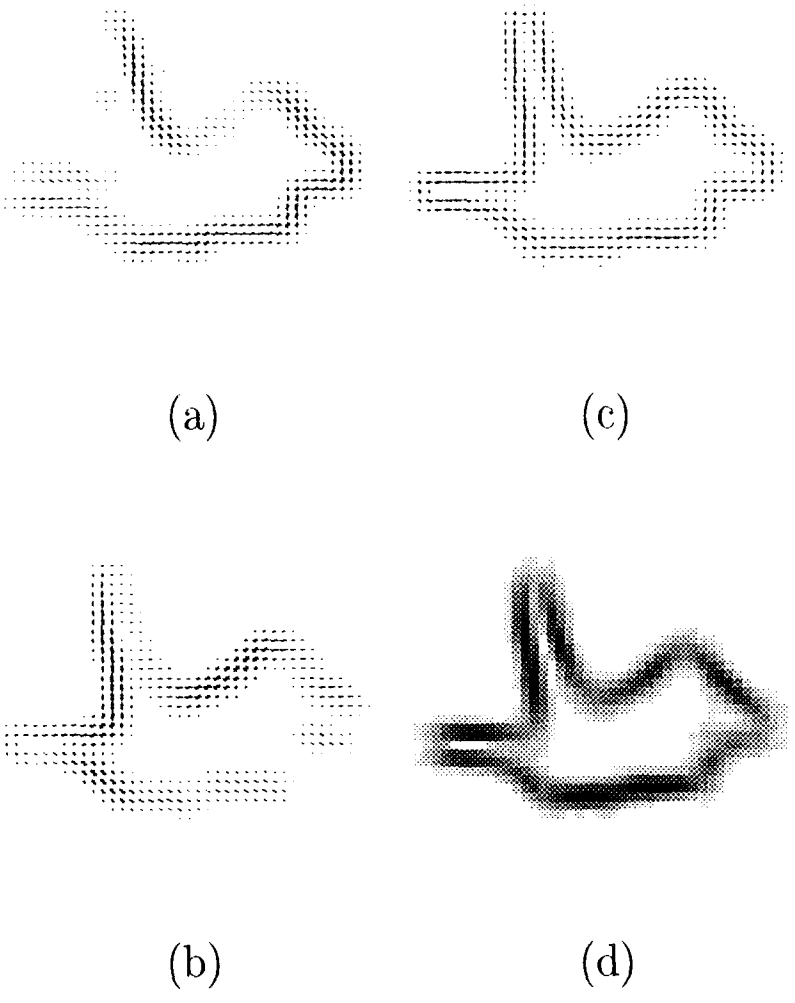

FIGURE 7. CORT-X 2 processing. (a) Output resulting from the ON-C network, using left sided elliptical filters (simple cell output) with a small spatial scale (6 × 3 pixels). A "left-sided" filter refers to filters that respond to a vertical left-to-right, high-to-low contrast transition area in the image when the filter is in vertical orientation. A "right-sided" filter is the opposite. Lines in the figure are proportional to the magnitude of the response at each orientation at each position. (b) Output from the OFF-C network using small left sided elliptical filters. (c) Hypercomplex cell output for the small scale. (d) The final CORT-X 2 output.

at both the small and large spatial scales (see Figure 5).

Complex cells excite hypercomplex cells in the next layer at their position and orientation while inhibiting hypercomplex cells at nearby locations that are not colinear with the complex cell's orientation. This positional competition is called the *first competitive stage*. It positionally sharpens the location of the segmentation, especially in response to textured and shaded images. Figure 7(c) shows the output of the hypercomplex cell layer for the small scale.

The next layer, called the *second competitive stage*, chooses the hypercomplex cell whose orientation is maximally activated to represent the activity at each position. These orientationally favored hypercomplex cells are often called higher-order hypercomplex cells in the full BCS. Figure 8(a) displays the small scale output of these higher-order cells.

The final stages of CORT-X 2 involve cooperative interactions between the large and small scale filters to join together the better boundary completion and noise suppression properties of the larger scale cells and the better localization properties of the smaller scale cells. One type of cooperative interaction is used to complete boundaries across boundary gaps caused by image noise. In the full BCS, boundary completion bridges gaps between distant image contrasts using a feedback loop that includes another cell type, called the bipole cell. Lacking a feedback loop, the CORT-X 2 filter uses a simplified interaction that captures the main heuristic for completing boundaries across small image gaps. In particular, cooperative interactions among the higher-order hypercomplex cells activate an inactive cell if enough cells that share the inactive cell's orientation are active on both sides of its oriented axis.

Another type of cooperative interaction combines large and small scales in such a way that the better

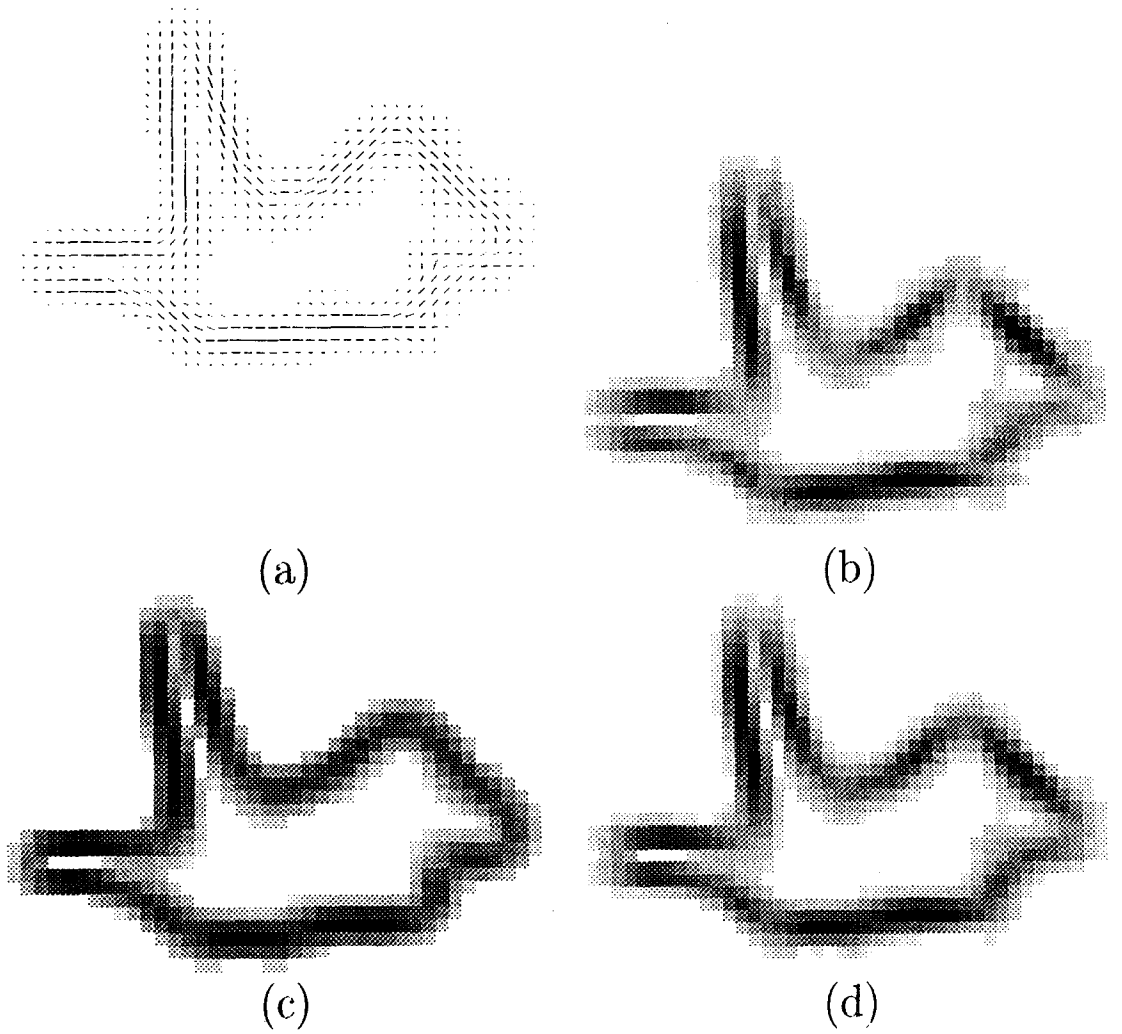

FIGURE 8. CORT-X 2 final state outputs. (a) The maximal orientations of the hypercomplex cells (second competitive stage). (b) The long-range boundary completion output. (c) The multiple scale interaction output. (d) The final CORT-X 2 output from the additive combination of the top right and bottom left outputs.

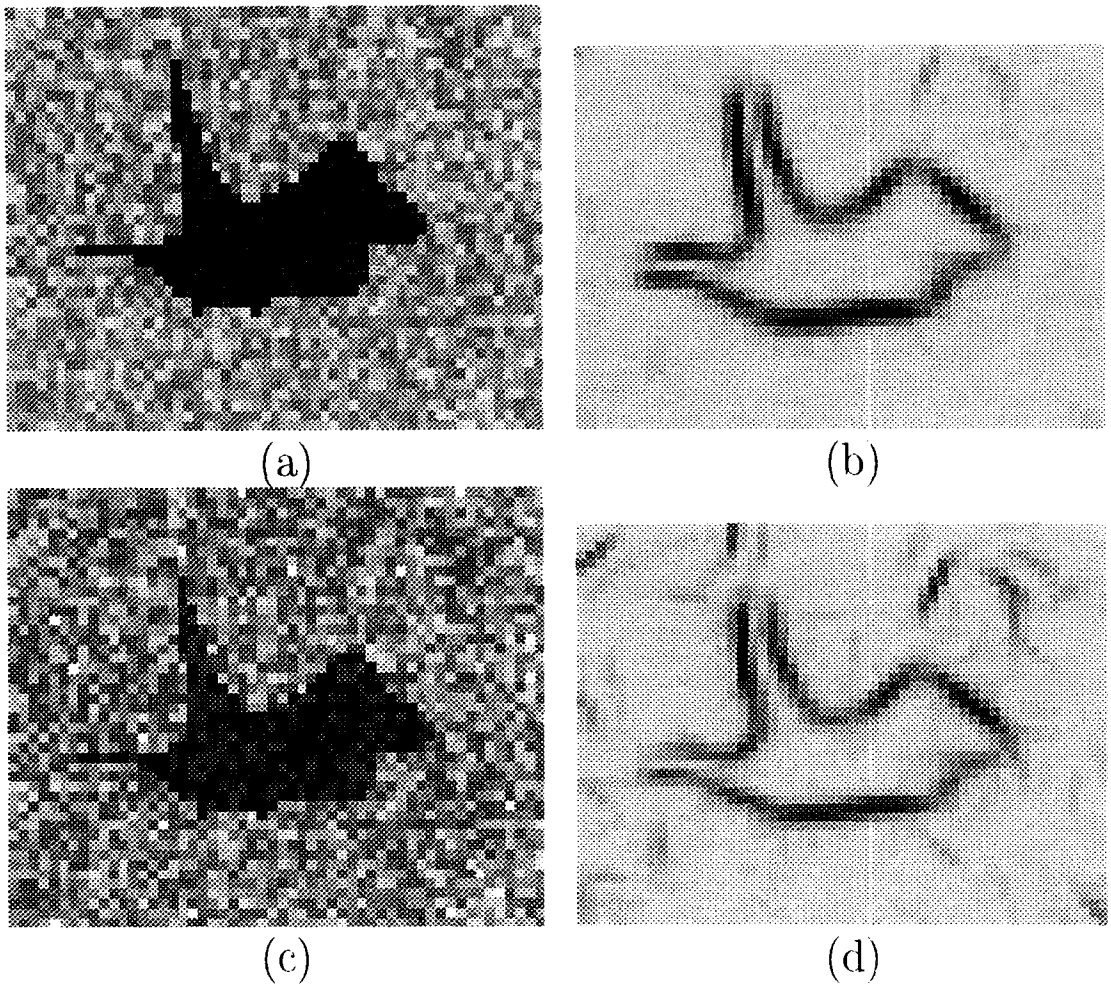

FIGURE 9. Results of processing noisy images with CORT-X 2. Uniform random noise was added to every pixel in the original image. The original image (left column) had pixels with activity levels between 0.0 and 1.0. Uniform random noise with pixel values ranging between 0.0 and 1.0 was scaled by C and added to the clean image prior to processing by CORT-X 2 with results shown in the right column. At top (a, b), random noise between 0.0 and 0.5 (C = 0.5 or 50% noise) was added, on the bottom (c, d), noise between 0.0 and 1.0 (C = 1.0 or 100% noise) was added. For example, 100% noise refers to the fact that the magnitude of any given noise pixel can be as large as the largest magnitude pixel in the noise-free image.

localization properties of the smaller scale filters have an effect only within regions where the larger scales have located a boundary. The output of the boundary-completing cooperative cells is shown in Figure 8(b). Figure 8(c) displays the multiple scale cooperative interaction, and Figure 8(d) shows the final CORT-X 2 output consisting of the sum of output of the boundary-completion and multiple scale localization interactions. Figure 9 shows the results of processing images with two levels of additive noise: $C=0.5$ or 50% noise (a), and $C=1.0$, or 100% noise (c). Because of the relative simplicity of the images being processed, only amplitude information was used. The CORT-X filter also computes potentially useful information about the relative orientation of different object parts, as in Figure 8(a). The boundary completion capabilities of the CORT-X filter are also not greatly taxed by these images.

4. TRANSLATION, ROTATION AND SCALE INVARIANCE

The next processing stages generate a representation, using standard methods, that is invariant under 2-D translation, rotation, and dilation. First, the 2-D boundary segmentation is centered by dividing its first moments by its zeroth moment to find the figure centroid, subtracting off the center of the image and then shifting the figure by this amount. A log-polar transform is then taken with respect to the center of the image. Each point (x, y) is represented as $re^{i\theta}$. Taking the logarithm yields coordinates of log radial magnitude and angle. As is well known (Schwartz, 1977), figural sizes and rotations are converted into figural shifts under log-polar transformation. Using these shift parameters to center the log-polar transformed image leads to a figural representation that is

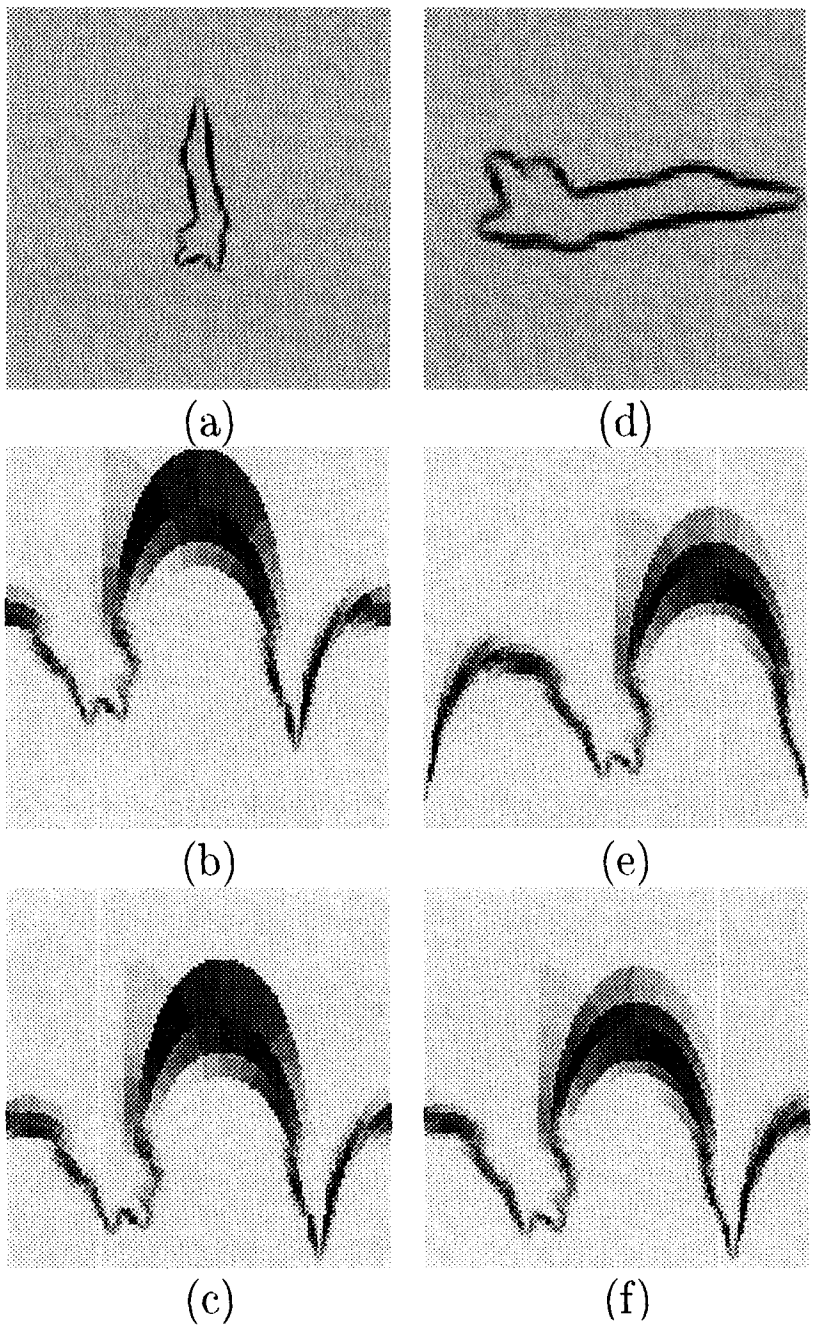

FIGURE 10. Log-polar transform example. At top (a,d) are the results of processing F-18 images at two different scales and orientations using CORT-X 2. The middle images (b,e) show the results of a log-polar transform of the top images. At bottom (c, f) are the centered log-polar images that have been made more identical in the sense that, overall, many more (*x*, *y*) pixel locations obtain similar values between the two images after processing. In the log-polar images, the ordinate is the log-radial magnitude and the abscissa is the periodic angle axis.

invariant under 2-D changes in position, size and rotation. Figure 10 shows the results of processing two F-18 images which are identical except for scaling and rotation. The images become very similar in the centered log-polar domain.

An alternative approach to invariance filtering, called the What-and-Where filter (Carpenter et al., 1992b, 1993a) could also have been used, and has distinct advantages for generalizing the VIEWNET architecture to scene understanding applications, as noted in Section 9.

5. COARSE CODING

Coarse coding reduces memory requirements, as it compensates for modest 3-D foreshortening effects and inaccuracies of figural alignment in the invariant filter. Coarse coding by averaging in the space

domain is equivalent to low pass filtering in the frequency domain. Neighboring pixel features are hereby blurred, compensating for slight alignment variations. In addition, 2-D images of 3-D objects suffer from 3-D perspective distortions that cannot be corrected by log polar transforms. These are viewpoint-specific foreshortening effects and self-occlusions. The blurring associated with coarse coding can help to increase generalization by causing foreshortened and nonforeshortened images to map to nearly the same image. On the other hand, too much blurring can obscure critical input features and thereby harm recognition performance. Our analysis suggests how to balance these effects to maximize the benefits of coarse coding.

Data reduction is an important practical issue in its own right in many realistic pattern recognition problems. Seibert and Waxman's database contains 4300 images of three distinct objects. Each image is made up of 128×128 floating point pixels of 4 bytes each for a total of $128 \times 128 \times 4 \times 4300 = 281{,}804{,}800$ bytes. Yet, as reported below, object identification performance based on single 2-D views did well even when images were reduced to just 4×4 pixels, yielding a database of just $4 \times 4 \times 4 \times 4300 = 275{,}200$ bytes. This reduction affords an enormous saving in both computation and memory storage.

Coarse coding of the 2-D images used a spatial averaging method that preserves sufficient information for accurate recognition. This method was selected as follows. Spatial averaging consists of convolving the original image I with a function Ψ and then sampling the resultant image with delta functions spaced every T pixels: $\delta(x - nT, y - kT)$. For simplicity, in 1-D this is

$$(I * \Psi) \cdot \sum_{n=-\infty}^{\infty} \delta(x - nT). \tag{1}$$

If the Fourier transform of I is $\hat{I}$, and that of Ψ is $\hat{\Psi}$, then the Fourier transform of equation (1) is

$$(\hat{I} \cdot \hat{\Psi}) * \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s), \tag{2}$$

where $\Omega_s = 2\pi/T$, and T is the sampling period in pixels. If Ω_N is the highest frequency in the image, then for the image to be uniquely determined by its samples, we must have by the Nyquist sampling theorem that

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_N. \tag{3}$$

Two simple spatial averaging functions Ψ are: (I) uniform averaging of the input image so that all pixels in a window of some width are summed and divided by the number of pixels in the window; (II) Gaussian averaging of the input image so that a normalized Gaussian weighted sum of all pixels is taken over a window of some width. Both approaches were investigated in this paper.

Method (I) was considered first, using a rectangular window centered on each sampling point nT of width $2T$. A problem with this approach is that a rectangular filter in space is a sinc function in frequency. The side lobes of the sinc function can introduce high frequency aliasing ("ringing") in the resultant image. High frequency ringing can be reduced by using a function that provides a better low pass filter. This is true of the Gaussian function of method (II), which has the further advantage of being an eigenfunction of a Fourier transform, since the Fourier transform of a Gaussian is a Gaussian with reciprocal variance, thereby simplifying calculation.

For Gaussian-based spatial averaging, we must determine how to best set the standard deviation σ of the Gaussians. Let us define two standard deviations away from the Gaussian midpoint to be essentially zero. The cutoff frequency of such a low pass filter is then $\pi/2\sigma$. From eqn (3), we must have

$$\frac{2\pi}{T} > \frac{2\pi}{2\sigma} \tag{4}$$

which yields at equality

$$\sigma = \frac{T}{2}. \tag{5}$$

By (5), the standard deviation should be set to half the Gaussian center-to-center sampling period so that the zero point of each Gaussian just touches the center of the next Gaussian as in Figure 11(c). Figure 11(d–f) shows the results of coarse coding the image in Figure 11(b) in this way to reduce the original image of 128×128 pixels down to 16×16, 8×8, and 4×4 pixels, respectively.

A third method of coarse coding could also be envisioned: truncating an infinite series expansion of the image with orthogonal filters. This approach is computationally burdensome unless one computes only the first few terms of the expansion. If sinusoidal filters are used and the series is truncated, then this method becomes equivalent to low pass filtering with a rectangular filter. Since a rectangular filter in the frequency domain is a sinc function in the space domain, this approach is equivalent to first convolving the original image with a sinc function. It would thus be desirable to use a smoother function with

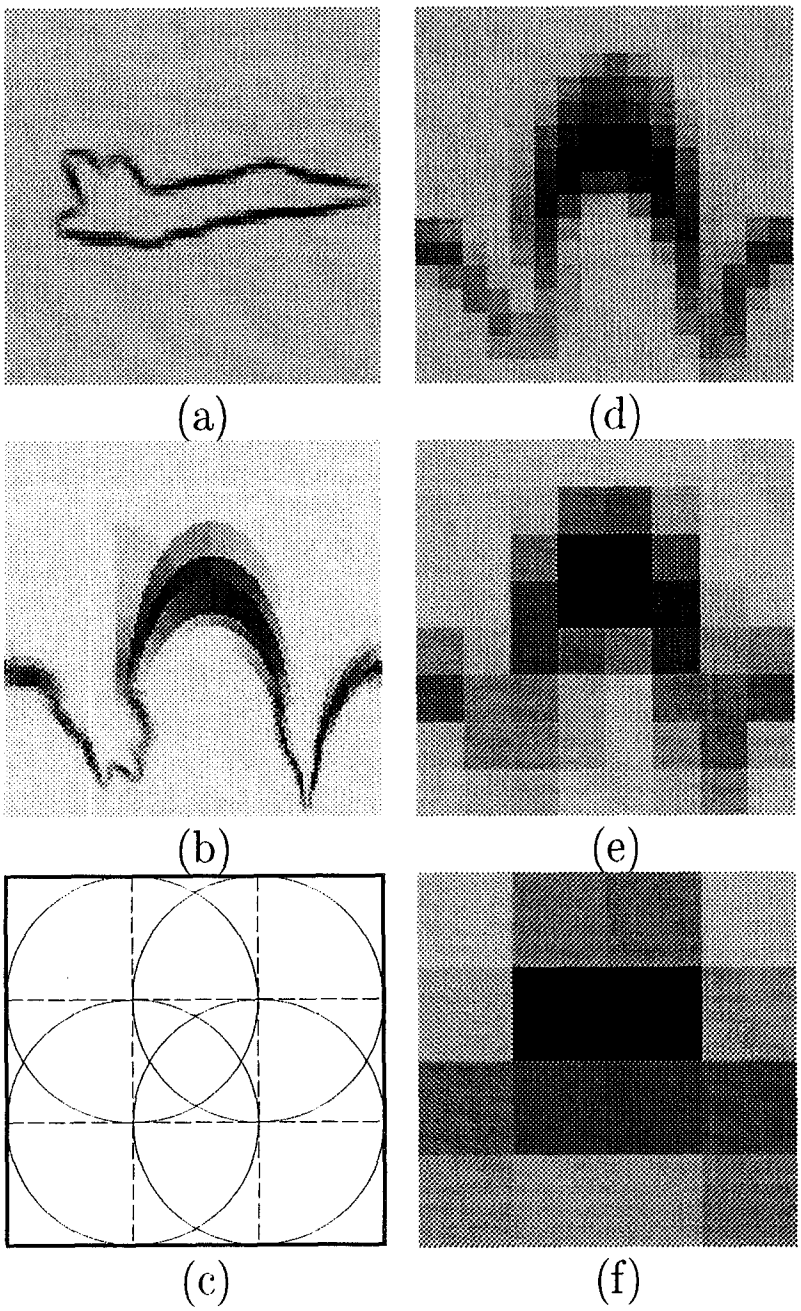

FIGURE 11. Preprocessing summary. (a) Output of CORT-X 2 preprocessing. (b) Centered log-polar image. (c) Gaussian coarse coding pattern. (d–f) Coarse coding reduction from 128 × 128 pixels down to 16 × 16, 8 × 8, and 4 × 4 pixels.

which to do the low pass filtering, but this is exactly what the Gaussian averaging of method (II) achieves.

6. LEARNING RECOGNITION CATEGORIES USING FUZZY ARTMAP

The Fuzzy ARTMAP architecture was used to learn 2-D view categories from the coarse-coded invariant spectrum of the noise-suppressed boundary segmentations. Fuzzy ARTMAP was chosen because it can achieve stable fast incremental learning of categories in response to unlimited amounts of nonstationary input data, can run in both unsupervised and supervised modes, and in its supervised mode can fit the size, number, and shape of its categories to the input statistics. Fuzzy ARTMAP parameters were chosen to enable the network to learn the conditional probability of the true 3-D object given the selected 2-D view category. We utilized the simplified version of the Fuzzy ARTMAP network of Carpenter et al. (1992a) that was employed in Carpenter et al. (1992c). This circuit consists of a Fuzzy ART module (Carpenter et al., 1991) ART_a that learns 2-D view categories and a field of 3-D object category

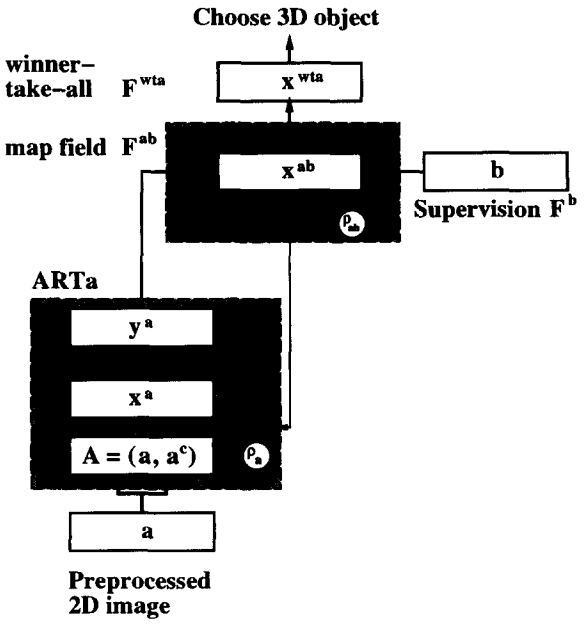

FIGURE 12. Fuzzy ARTMAP architecture. Each preprocessed 2-D input vector a is fed sequentially to the network as it becomes available. The inputs are complement coded which transforms the *M*-vector a into the 2*M*-vector A = (a, a^c) at field F_0^a which is then fed into the input field F_1^a. A category node k is chosen at F_2^a which reads out its prediction to the Map Field via weights w_k^{ab}. If the prediction is disconfirmed, a match tracking process is invoked in ART_a. Match tracking raises the ART_a vigilance ρ_a to just above the match ratio $|x^a|/|A|$. This triggers an ART_a search which activates either a different existing category, or a previously uncommitted category node at F_2^a. After the search process concludes, F^{wta} chooses the maximally activated node in F^{ab} as the 3-D object being viewed.

output nodes F^b. The 2-D view and 3-D object category nodes are linked together by an associative memory F^{ab} that is called the *Map Field* (Figure 12). In supervised learning mode, Fuzzy ARTMAP receives a sequence of input pairs $(\mathbf{a}_p, \mathbf{b}_p)$ where $\mathbf{b}_p$ is the correct 3-D object class given the analog 2-D view input pattern $\mathbf{a}_p$. The ART_a module classifies analog input vectors $\mathbf{a}_p$ into categories and the Map Field makes associations from the ART_a categories to the outputs $\mathbf{b}_p$ in F^b. This simplified architecture thus does not include an ART_b module for independently clustering the outputs $\mathbf{b}_p$ into their own categories.

Under supervised learning conditions, if $\mathbf{a}_p$ is categorized into an ART_a category that predicts an incorrect output $\mathbf{b}_p$, then the mismatch between actual and predicted $\mathbf{b}_p$ causes a memory search within ART_a via the *match tracking* mechanism. Match tracking raises the ART_a vigilance parameter ρ_a by the minimum amount that will trigger a memory search. In particular, ρ_a grows until it just exceeds the match value between the input vector $\mathbf{a}_p$ and the prototype of the active ART_a category. Since low vigilance leads to learning of large, coarse categories and high vigilance leads to learning of small, fine categories, match tracking sacrifices the minimum amount of category compression that is needed to correct each predictive error. Memory search by match tracking continues until a pre-existing ART_a category that predicts the correct ART_b category is found, or a new ART_a category is chosen.

After one of these conditions is satisfied, learning takes place both within ART_a and from the chosen ART_a category to the Map Field. Match tracking assures that predictive error is minimized while maintaining maximum generalization during fast or slow incremental learning conditions. Between learning trials, vigilance relaxes to its baseline vigilance $\bar{\rho}_a$. In test mode, input vectors $\mathbf{a}_p$ are classified by ART_a and the chosen category reads out its prediction to the Map Field. The index of the maximally activated node in the Map Field is taken to represent the predicted output class.

The input vector $\mathbf{a}_p$ is *complement coded* before it activates ART_a. This preprocessing step enables the network to code both input features that are critically present and input features that are critically absent. In response to an input vector $\mathbf{a}_p$, complement coding delivers an input vector $\mathrm{A}_p = (\mathbf{a}_p, \mathbf{a}_p^c)$ to ART_a, where $\mathbf{a}_p^c = \mathbf{1} - \mathbf{a}_p$. Complement coding also normalizes the total input A_p to ART_a such that $\|\mathrm{A}_p\|_1 = 1$. It thereby prevents a category proliferation problem that could otherwise occur (Carpenter et al., 1991). Complement coding means intuitively that an input vector turns ON the cells corresponding to $\mathbf{a}_p$ as it turns OFF the cells corresponding to $\mathbf{a}_p^c$, much as in the ON and OFF channels of the CORT-X 2 filter. The algorithm is mathematically defined in Appendix B.

Fuzzy ARTMAP parameters were chosen to allow for on-line slow learning from ART_a F_2^a to the Map Field nodes. A maximal ART_a vigilance level, $\bar{\rho}_{max}$ was introduced such that an error at the Map Field triggers match tracking only if match tracking leads to a vigilance $\rho_a \leqslant \bar{\rho}_{max}$. This bound prevents categories from becoming too small. In response to an error that would otherwise cause ρ_a to exceed $\bar{\rho}_{max}$, learning takes place instead from the active node in F_2^a to the Map Field. By setting the Map Field learning rate β_{ab}, baseline $(\bar{\rho})$ and maximal $(\bar{\rho}_{max})$ vigilance levels appropriately, weights from F_2^a nodes to the Map Field may begin to approximate the conditional probability of the true class (the 3-D object) given the selected F_2^a category (the 2-D view category). A related approach to slow probability learning is described in Carpenter et al. (1993b).

7. SIMULATION RESULTS

A computer simulation was run on the airplane database using the CORT-X 2 parameters shown in Table 1. The database was processed twice by CORT-X 2, once with a large pair of large and small oriented filters and once with a smaller pair of large and small

filters. The large oriented filter pairs consisted of elliptical receptive fields with axes 16 × 8 and 10 × 5 pixels. The small oriented filter pair consisted of oriented ellipses with axes 10 × 5 and 6 × 3 pixels. This was done so that recognition results could be compared when images were processed at different scales. Coarse coding was done with both simple spatial averaging and Gaussian averaging, reducing the image down to 16 × 16, 8 × 8, and 4 × 4 pixels from an original size of 128 × 128. The window for simple spatial averaging was square with a width twice the sampling period T; that is, a window centered at one pixel extended until it just touched its neighboring pixels. The standard deviation for Gaussian averaging was set to $T/2$, as discussed in Section 5. Training and testing sets were assembled as discussed in Section 2 with even numbered images forming the training set and odd numbered images forming the testing set. Except where explicitly mentioned, the simulations were run with the parameters shown in Tables 1 and 2.

The data were presented to the network in two different ways: (1) 2-D views were presented in the "natural" order in which they would appear if viewing the actual object in motion; (2) 2-D views were presented in random order. These two methods of data presentation were used to test whether presenting views in natural order helps recognition scores. Training in natural order consisted of 160 runs of from 1 to 50 views over each object. Training in random order consisted of a series of 40 runs of 100 training set views over each object. Recognition scores are taken as an average of fifteen separate training-test cycles.

TABLE 1
The Parameter Set Used for CORT-X 2 in the Simulations

Parameter	Description
$\kappa_{onC} = 7.0$	On-center magnitude
$\alpha_{onC} = 1.3$	On-center standard deviation
$\kappa_{offS} = 3.333$	Off-surround magnitude
$\alpha_{offS} = 1.875$	Off-surround standard deviation
$\bar{D} = B = 1.0$	Shunting values
$\bar{B} = D = 0.5$	Shunting values
$S = 0.2$	Spontaneous activity level
$A = 134$	Shunting decay
$\alpha_1 = \alpha_2 = 1.1$	Threshold contrast parameters
$\beta_1 = \beta_2 = \beta = 0.003$	Threshold noise parameters
$F = 0.5$	Complex cell scaling constant
$\varepsilon = 0.1$	Hypercomplex cell divisive offset
$\mu = 5.0$	Hypercomplex cell convolution scaling
$\tau = 0.004$	Hypercomplex cell threshold
$\delta = 0.001$	Long range cooperation threshold
$\pi/8$	Oriented kernel orientation spacing
$(a_2, b_2)_{large} = (16, 8)$	Large set, large ellipse axis
$(a_1, b_1)_{large} = (10, 5)$	Large set, small ellipse axis
$(a_2, b_2)_{small} = (10, 5)$	Small set, large ellipse axis
$(a_1, b_1)_{small} = (6, 3)$	Small set, small ellipse axis
$G_1 = 2a_1/3$	Hypercomplex small kernel diameter
$G_2 = 2a_2/3$	Hypercomplex large kernel diameter
$U = 2a_2/5$	Multiple scale interaction kernel diameter
$O = 3a_2/5$	Long-range cooperation kernel length

TABLE 2
The Fuzzy ARTMAP Parameter Set Used for the Simulations

Parameter	Description
$\alpha = 0.6$	Fuzzy ART search order
$\beta_a = 1.0$	Fuzzy ART learning rate
$\beta_{ab} = 1.0$	Map Field learning rate
$\bar{\rho} = 0.1$	Baseline Fuzzy ART vigilance ρ_a
$\bar{\rho}_{max} = 1.0$	Maximum ART_a vigilance
$\rho_{ab} = 1.0$	Map Field vigilance

7.1. Fast Learning Without Noise

No clear advantage results from ordered presentation as compared to unordered presentation using noise-free data ($C = 0$) and fast learning, as shown by the results in Table 3. It can be seen that the smaller CORT-X 2 filter set resulted in better recognition performance overall and did better given more detail (less coarse coding).

Figures 13 and 14 analyze an example of a recognition error. In Figure 13, the left and middle columns show two different views of an F-16 in sequence. The left view activated ART_a category 26 which was correctly associated with F-16 in the Map Field. The middle F-16 image activated ART_a category 28 which is associated with HK-1, resulting

TABLE 3
Recognition Results on a Noise-free Database (C = 0) CORT-X 2 Filter Sizes Refer to the Size of the Oriented Receptive Field Filters

		Coarse code using spatial avg/ Gaussian avg		
CORT-X 2 filter set	Data presentation	4 × 4	8 × 8	16 × 16
Small	Ordered	81.0/83.1	84.4/86.4	86.7/90.5
Small	Unordered	80.3/83.9	84.9/86.5	86.8/89.3
Large	Ordered	76.8/78.7	79.0/81.6	79.1/80.1
Large	Unordered	77.4/79.7	80.5/81.5	77.1/80.5

CORT-X 2 was run twice using a larger and a smaller set of its large and small elliptical oriented filters. In the table, "large" refers to the run with the larger set of oriented ellipses with axes 16 × 8 and 10 × 5 pixels; "small" refers to the run with the smaller set of oriented ellipses with axes 10 × 5 and 6 × 3 pixels. Views were presented either in natural order or in random order. Data was coarse coded from 128 × 128 down to 4 × 4, 8 × 8, or 16 × 16 using simple spatial averaging or Gaussian averaging. Recognition scores refer to the percent of 2-D views correctly associated with a 3-D object.

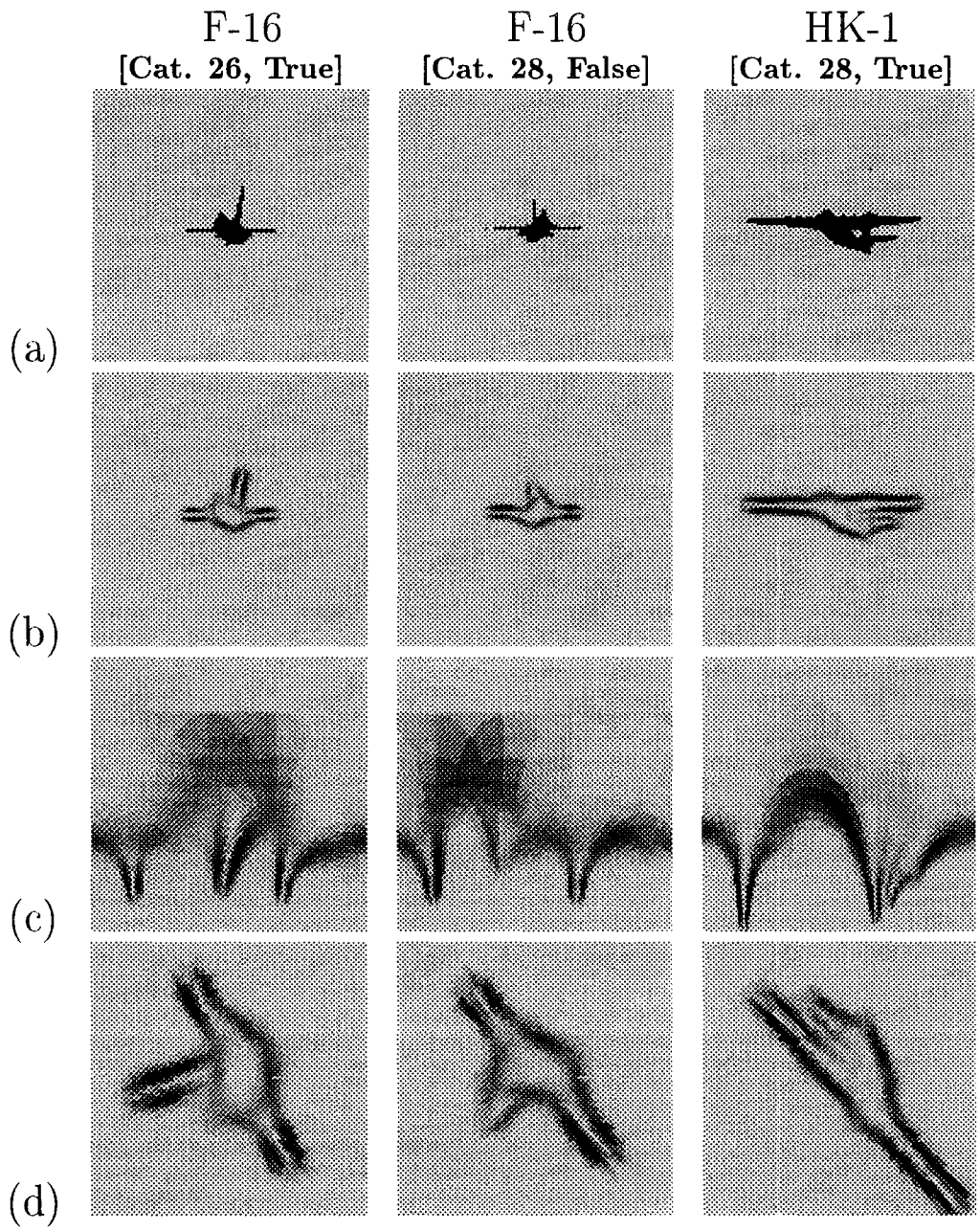

FIGURE 13. Analysis of an error in categorization I: The left and the middle column show two images of the F-16 jet in sequence. The F-16 image in the left column activated ART_a category 26 (y_{26}^a) which is associated with the Map Field node representing the class of F-16 jets. The F-16 image in the middle column activated ART_a category 28 (y_{28}^a) which is associated with the Map Field node representing the class of the HK-1. The right column shows one of the HK-1 images that correctly activates ART_a category 28. Thus, the left and right images were correctly recognized, the middle F-16 image was incorrectly recognized as an HK-1. (a) Shows the raw jet images. (b) Shows the results of CORT-X 2 processing with the larger set of filters and parameters given in Table 1. (c) Shows the results after a log-polar transform followed by centering. (d) Shows the inverse log-polar transform of the images in (c) where the confusion of the middle image with the right image becomes more apparent.

in a recognition error. The right column shows an HK-1 image typical of those which correctly activate ART_a category 28. Figure 13(a) shows the raw images, Figure 13(b) shows the images after CORT-X 2 filtering with the large scale set of filters, and Figure 13(c) shows the results of a centered, log-polar transform of the image. Figure 13(d) shows the inverse of the images in Figure 13(c) so that the confusion between the middle and the right columns becomes more apparent—quite a bit of the two images overlap.

The coarse coded representations of the invariant spectra show the confusion more clearly. Figure 14(a) shows the results of Gaussian coarse coding the log-polar images in Figure 13(c) followed by complement coding where the top half of each image is the normal

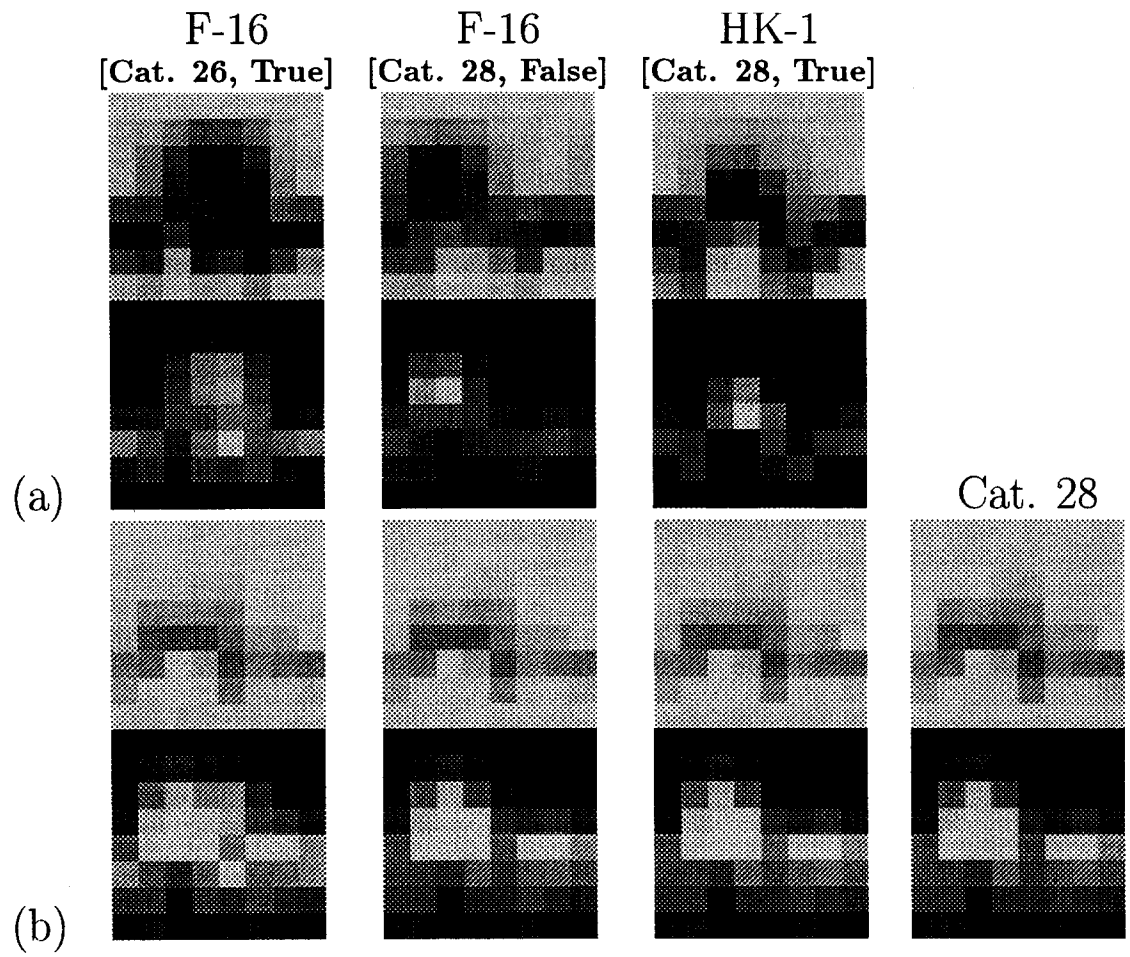

FIGURE 14. Analysis of an error in categorization II: (a) Shows images from Figure 13(c) which have been Gaussian coarse coded down to 8 × 8 pixels and then complement coded for input into the Fuzzy ARTMAP module. The bottom half of each image is the complement code of the top half in accordance with eqn (32). (b) Shows the fuzzy minimum activations of the three inputs of columns (1–3) in (a) with the weight vector in column four, e.g. $A_{(1-3)} \wedge w^a_{28}$. Note that the left input caused stronger activations (not shown) to category 26. The weights of ART_a category 28 are shown at far right. The F-16 image input at left caused category 26 to be chosen which in turn correctly activates the F-16 node in the Map Field. The F-16 input in the middle image caused category 28 to be chosen which activates the node for the HK-1, causing an error in recognition in this case. An HK-1 input that correctly activates category 28 is shown in the third column. It can be seen that the second column and third column activations are a perfect match (are a fuzzy subset) with the weights w^a_{28} in the fourth column, but the activation in the first column differs in the complement coded response. In this figure, darker shades indicate higher activations.

code and the bottom half is the complement code as in eqn (32). It can be seen again that the F-16 (middle) and HK-1 (right) coarse-coded images are more similar to one another than to the F-16 (left) image. In these images, darker shades represent higher activities. In Figure 14(b), the weights of ART_a category 28 are shown at right. This category nodes for HK-1 in the Map Field, but was also activated by the F-16 image in the middle column. The three column images in Figure 14(b) show how each image in Figure 14(a) activates the weights of category 28 according to eqn (34), which computes a normalized minimum that is closely related to the measure of fuzzy subsethood (Carpenter et al., 1992a). The patterns for the noncomplement coded input (top halves of Figure 14(b)) are all nearly identical, meaning that they all lie above the template in value. The problem shows up in the complement coded part of the input (bottom halves of Figure 14(b)) where the left image is clearly different from the nearly identical middle and rightmost images. This problem was caused by combining loss of too much detail by the larger set of CORT-X 2 filters with rotation by the log-polar transform of the HK-1 and the middle F-16 until they maximally overlapped. These coarse coded patterns were combined with the weights of the more general (larger fuzzy set) HK-1 category via the fuzzy AND operation [see eqn (34)]. The resulting values were not sufficiently different to choose different categories for the two coarse coded versions of the segmented and rotated spectra.

TABLE 4
Percent of Additive White Noise Surviving Processing by CORT-X 2 and Coarse Coding

	% Noise surviving CORT-X 2 filtering and coarse coding:	
	Large CORT-X 2 filters (16 × 8, 10 × 5)	Small CORT-X 2 filters (10 × 5, 6 × 4)
	1.79	2.42
	After Gaussian coarse coding from 128 × 128 down to:	
16 × 16	0.33	0.34
8 × 8	0.23	0.29
4 × 4	0.19	0.26
	After spatial average coarse coding from 128 × 128 down to:	
16 × 16	0.40	0.40
8 × 8	0.28	0.30
4 × 4	0.21	0.28

7.2. Fast Learning Simulation With Noise

The system was next tested with noisy data using additive white noise scaled by $C = 1.0$; that is, each pixel could have noise added to it less than or equal to the value of the maximum activity of pixels (= 1.0) in the original image. Table 4 shows what percent of the additive noise survives processing by CORT-X 2 alone, and by CORT-X 2 and coarse coding together, for two different filter sets and three different coarse codings. The percent noise surviving these transformations was measured by the following formula:

$$\max_{\forall(x,y)}\left[\frac{\Psi(\mathbf{I}+\mathbf{N}) - \Psi(I)}{C}\right] \times 100, \qquad (6)$$

where $\mathbf{I}$ is the image, $\mathbf{N}$ is the noise image, Ψ is the CORT-X 2 filter, $C > 0$ is the noise scaling parameter and (x, y) is the pixel index in the images. Table 4 represents the average results from ten measurements using eqn (6).

It can be seen that the combination of CORT-X 2 filtering followed by coarse coding is effective in reducing additive noise. Using a fast learning paradigm, the recognition results shown in Table 5 were similar to those for the noise-free case in Table 3, except for some minor falling off of recognition scores at the lowest level of coarse coding (the 16 × 16 case). Less coarse coding has the same effect on noise as raising the cutoff frequency of a low pass filter. Thus, as seen in Table 4, more noise gets through with less coarse coding, yielding slightly lower recognition performance.

TABLE 5
Recognition Results on Noisy Data ($C = 1$) with Fast Learning ($\beta_{ab} = 1.0$)

CORT-X 2 filter set	Data presentation	Coarse code using spatial avg/Gaussian avg: 4 × 4	8 × 8	16 × 16
Small	Ordered	80.1/83.3	84.5/85.9	84.2/89.1
Small	Unordered	79.4/83.2	83.9/86.4	84.3/88.0
Large	Ordered	76.6/79.4	79.3/80.8	75.8/79.3
Large	Unordered	76.0/79.7	78.4/80.7	75.5/79.0

These results differ little from the noise-free results in Table 3 (no noise condition) with the exception of some consistent reduction in scores for the 16 × 16 coarse coding.

Table 6 shows the number of nodes created by the network after training for the no-noise (left entry) and noise (right entry) results reported above. Noise causes a small increase in the number of categories formed on average as the network attempts to correct a greater number of noise-induced errors during supervised training.

TABLE 6
Average Number of ART_a Categories Formed During Training for the Simulations of Table 3 (No-noise) and Table 5 (Noise)

CORT-X 2 filter set	Data presentation	Coarse code using spatial avg/Gaussian avg: 4 × 4	8 × 8	16 × 16
Small	Ordered	[172, 184]/[165, 169]	[77, 73]/[70, 73]	[34, 33]/[33, 35]
Small	Unordered	[191, 198]/[175, 179]	[76, 77]/[73, 76]	[34, 35]/[35, 36]
Large	Ordered	[168, 179]/[160, 162]	[71, 68]/[67, 71]	[31, 33]/[30, 31]
Large	Unordered	[183, 192]/[169, 174]	[73, 75]/[69, 72]	[32, 32]/[33, 32]

The format in the table is as follows: [spatial avg]/[Gaussian avg] = [no − noise, noise]/[no − noise, noise].

TABLE 7
Recognition Results on Noisy Data ($C = 1$) with Slow Learning to the Map Field ($\beta_{ab} = 0.2$, $\bar{\rho}_{max} = 0.95$)

CORT-X 2 filter set	Data presentation	Coarse code using spatial avg/ Gaussian avg 4 × 4	8 × 8	16 × 16
Small	Ordered	79.9/83.1	84.0/85.6	84.7/89.9
Small	Unordered	78.8/83.3	83.2/85.7	84.9/89.1
Large	Ordered	76.3/78.2	78.5/81.5	77.0/78.8
Large	Unordered	77.4/80.2	79.6/80.41	75.8/79.2

Due to the low levels of noise surviving preprocessing, the recognition results here are not substantially different than those found using fast learning in noise in Table 5 except where noise was highest as in the 16 × 16 coarse coding. As noise increases, slow learning becomes more important for maintaining good recognition scores.

7.3. Slow Learning Simulation With Noise

For the final set of computer simulations, the network was run on the noisy data using slow learning to the Map Field [$\beta_{ab} = 0.2$ in eqn (44)]. Fast learning was still used within the ART_a module itself however [$\beta_a = 1.0$ in eqn (43)]. In addition, a maximum vigilance level in ART_a ($\bar{\rho}_{max} = 0.95$) was set so that when match tracking due to error feedback attempts to create an ART_a category smaller than a given size, no new category forms and learning takes place for the current category instead. With noisy inputs, rather than continually making new categories to correct for the noise, ART_a categories below a set prescribed size begin to learn the conditional probability of the true class (the 3-D object) given the selected ART_a category (the categorical 2-D view).

Note that for $\bar{\rho}_{max} = 1.0$, the results for slow learning and fast learning (Section 7.2) to the Map Field are equivalent. They are equivalent because, with Map Field vigilance set to $\rho_{ab} = 1.0$ as in Table 2, the slightest mismatch at the Map Field will invoke match tracking and a new category will be created. The main difference between slow and fast learning using $\bar{\rho}_{max} = 1.0$ is that ART_a categories may learn their associations to nodes in the Map Field at different rates. The weights from an F_2^a node in ART_a to the correct node in the Map Field will always have a value of 1.0, however, since any error is corrected by forming a new category. Weights to the other nodes in the Map Field will be less than 1.0 (slow learning) or equal to 0.0 (fast learning). Recognition results on the test set are not hereby affected, since a winner-take-all field chooses the maximum activation in the Map Field as the recognition code via eqn (45) in Appendix B.

To derive benefit from slow learning, in the case $\rho_{ab} = 1.0$, we set $\bar{\rho}_{max} = 0.95$. For $\rho_{ab} = 1.0$, we may then compare the results of fast learning to the Map Field using $\bar{\rho}_{max} = 1.0$ with the results of slow learning to the Map Field using $\bar{\rho}_{max} = 0.95$. Table 7 records the results using slow learning in large amplitude noise ($C = 1$). Where noise levels after preprocessing were very small, the results were approximately the same as in the fast learning case shown in Table 5. Slow learning begins to help when the noise level increases, as with the 16 × 16 coarse coding. Table 8 records the average number of categories formed for the noisy data case using fast learning and slow learning. Slow learning with $\bar{\rho}_{max} = 0.95$, caused approximately 10% fewer categories to be formed than with $\bar{\rho}_{max} = 1.0$, since noise-induced errors do not always cause the formation of a new category in the former case.

8. EVIDENCE ACCUMULATION BY VOTING OR VIEW TRANSITIONS?

For a recognition system that can gather information from successive 2-D views, a key question is: given that an error occurs, how many successive errors will follow on average? For the airplane data set as processed by VIEWNET, it was found that the average overall length of an error sequence was 1.31 2-D views with a standard deviation of 0.57 views. On average then, when an error occurs, collecting two more views will usually be sufficient to correct the error. Thus, as in Seibert and Waxman's system, better 3-D object predictions may be derived by

TABLE 8
Average Number of Nodes Formed During Training for the Simulations of Tables 5 (Noise with Fast Learning) and 7 (Noise with Slow Learning)

CORT-X 2 filter set	Data presentation	Coarse code using spatial avg/Gaussian avg 4 × 4	8 × 8	16 × 16
Small	Ordered	[184, 165]/[169, 150]	[73, 67]/[73, 66]	[33, 30]/[35, 32]
Small	Unordered	[198, 180]/[179, 163]	[77, 69]/[76, 70]	[35, 32]/[36, 33]
Large	Ordered	[179, 160]/[162, 147]	[68, 61]/[71, 66]	[33, 30]/[31, 29]
Large	Unordered	[192, 175]/[174, 160]	[75, 69]/[72, 67]	[32, 30]/[32, 30]

The format in the table is as follows: [*spatial avg*]/ [*Gaussian avg*] = [*fast learning, slow learning*]/[*fast learning, slow learning*]. It can be seen that slow learning reduced the number of nodes formed by approximately 10%.

accumulating evidence from 2-D views. This is accomplished in VIEWNET in perhaps the simplest way by using a working memory whose unordered states are updated whenever a new 2-D view category is chosen. The working memory is realized as an integration field (F_k^{int}) between the Map Field (F) that codes 3-D object categories and the winner-take-all field (F^{wta}) in Figure 12. The equation for the integrator field is updated each time ART_a chooses a new 2-D view category:

$$(x_k^{int})^{new} = \beta_{int} x_k^{ab} + (1 - \beta_{int})(x_k^{int})^{old}, \qquad (7)$$

where x_k^{int} is an integrator node for the kth 3-D object, β_{int} is the integration rate each time the equation is stepped, and x_k^{ab} is the kth Map Field 3-D object category. The integration node with the largest activation represents the prediction of the 3-D object being viewed by the system. This maximum activation in the integration field is chosen by the winner-take-all field (F^{wta}) as the network's prediction of the 3-D object.

Figure 15 shows a simulation of the integrator field where eqn (7) is stepped once each time ART_a chooses a category. Three integrator nodes are shown along the ordinate, one for each of the airplanes. Grey shading in the figure shows which 3-D object is "winning". In this simulation, the 3-D object being viewed is an F-16. The sequence of ART_a F_2^a categories that occurred is shown along the abscissa. The first two categories, 1 and 22 were in error since they were associated with the F-18 node in the Map Field. Categories 21 and 26 code correctly for the F-16 followed by category 48 which codes for the HK-1. Next, category 71 is selected. It is an uncommitted category that has never been activated before. By default, it gives equal activation to all integrator nodes. The remaining categories code correctly for the F-16.

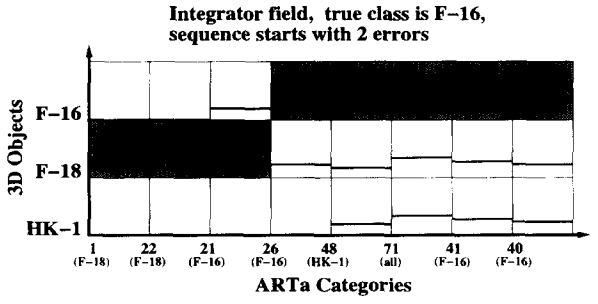

FIGURE 15. Simulation of the integration field. The ordinate axis contains the integrator nodes representing the 3-D objects: F-16, F-18, and HK-1. The abscissa represents ART_a F_2^a categories chosen by the preprocessed 2-D views being presented to VIEWNET. Black horizontal lines denote activity of the corresponding integrator node through time. Gray shading in the figure indicates the integrator node with maximal activation which represents the VIEWNET decision as to which object is being presented. Categories 1 and 22 erroneously code for the F-18 here. Category 48 erroneously codes for the HK-1. Category 71 has not been chosen before and so selects all objects simultaneously. The rest of the categories code correctly for the F-16. The integration step size was set to $\beta_{int} = 0.2$.

Implementing evidence accumulation in this way is similar to voting for the 3-D object over a sequence of 2-D view inputs, but with recent views being given more weight. For $1 \geqslant \beta_{int} \geqslant 0$, the closer β_{int} is to one, the more weight is given to recent votes. To measure performance on the test set with voting, the integrator field was allowed to collect first two (for the two votes score), or three (for the three votes score) activations before VIEWNET's recognition decision was recorded. The integrator field was then cleared and two or three more activations were again collected before the next decision was made. This process was repeated until 1000 views had been seen in the test set for each object at which time the percent correct recognition score was computed.

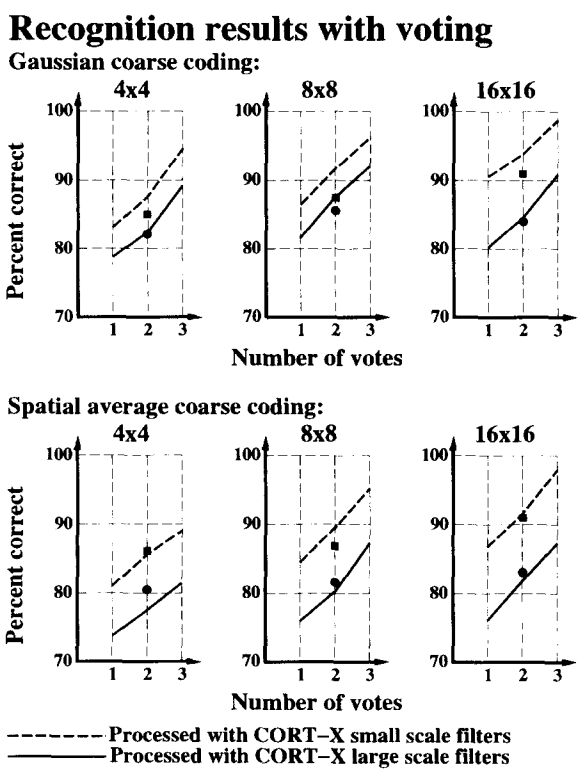

FIGURE 16. Recognition results for voting with an integration rate of $\beta_{int} = 0.2$. The graphs show the recognition results after gathering evidence over one, two and three 2-D views for data preprocessed using large (solid line) and small (dotted line) scale CORT-X 2 filters. Results from both Gaussian and spatial averaging coarse coding methods are shown where the images were reduced from 128 × 128 down to 4 × 4, 8 × 8 and 16 × 16 pixels. The circles and squares represent recognition scores resulting from using view transitions as discussed in Section 8.1. The black circles represent the recognition scores using view transitions for preprocessing with the large scale CORT-X 2 filters, the black squares represent recognition scores using view transitions for preprocessing with small scale CORT-X 2 filters.

Figure 16 shows the average recognition scores for voting with $\beta_{int} = 0.2$ over one, two, and three views under CORT-X 2 preprocessing with large- and small-scale filter sets and coarse coding to 4×4, 8×8 and 16×16 pixels using both Gaussian and spatial averaging. Voting over three frames improves recognition results by an average of 10% with the best results being 98.5% correct for small-scale filtered, 16×16 Gaussian coarse-coded data. The black dots and squares in the figure show recognition results from using 2-D view transition information as explained below.

8.1. Do View Transitions Lead to Better Recognition?

The advantage of voting over using 2-D view transitions is that, given N 2-D views, the $O(N^2)$ cost for learning view transitions is avoided. To compare how well voting over view sequences does relative to using view transitions, an architecture was simulated that incorporates view transition information into 3-D object recognition. Figure 17 shows such a modified VIEWNET 2 architecture in which a working memory is installed above the ART_a module, and another Fuzzy ART module, ART_s, categorizes the sequential information represented by the working memory and makes associations to 3-D object nodes in the Map Field.

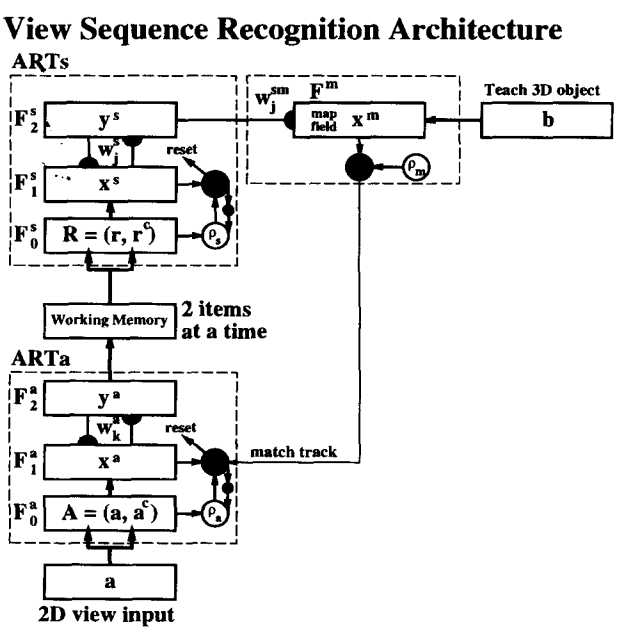

FIGURE 17. VIEWNET 2 architecture modified to learn 3-D object categories from 2-D view category sequences that are sequentially stored in working memory. ART_a categorizes 2-D views which are then input into a working memory called STORE that encodes the order in which items entered memory in an activation gradient. The working memory only keeps two items active in a spatial pattern where activation levels code for order of item entry into the memory. In this way, two different activations in working memory represent a transition from one ART_a categorical 2-D view to another. The pattern of activations is learned by ART_s which takes the working memory activations as input after complement coding them. ART_s then associates its learned categorical 2-D view transitions with the 3-D object represented in the Map Field. This architecture was used to compare using view transitions versus using evidence accumulation over two views for 3-D object recognition.

The working memory that we use transforms the temporal sequence of categorical activations y^a in ART_a layer F_2^a into an evolving spatial pattern of activation. This type of working memory does not explicitly code view transitions. Rather, it explicitly codes the 2-D view categories in terms of which working memory nodes are active (the "item" information), and implicitly codes view transitions in terms of their relative activity (the "order" information). Such "item and order" working memories are suggested by a variety of cognitive data. Grossberg (1978a, b) developed an explanation of such data as emergent properties of a design that enables a self-organizing feature map or ART system to stably learn to categorize item and order information as new items continue to be stored. Bradski et al. (1992, 1994) showed how to realize such a working memory design in a class of neural networks called STORE (Sustained Temporal Order REcurrent) models.

In the present application, the model is designed to store only two active views at a time to simulate encoding of the temporally ordered transition from one categorical ART_a view to another. The following system of STORE model equations implements this design goal

$$\frac{dx_i}{dt} = [AI_i + y_i - x_i x - Bx_i]I, \tag{8}$$

and

$$\frac{dy_i}{dt} = [x_i h[x_i - \Gamma] - y_i]I^c, \tag{9}$$

where x_i are y_i are activities of nodes in the bottom and top layers, respectively, of the STORE working memory, I_i is the ith input into the working memory, $x \equiv \sum_k x_k$, $I \equiv \sum_k I_k$, $I^c \equiv 1 - I$, and $h[x_i - \Gamma] = 1$ if $x_i - \Gamma > 0$, else $h[x_i - \Gamma] = 0$. To keep only two items active at a time, parameters were set to $A = 1.1$, $B = 0.5$, and $\Gamma = 0.3$.

Fuzzy ART module ART_s receives inputs from the STORE model and makes associations to the Map Field nodes, which represent the possible 3-D objects being viewed. Match tracking operates only on ART_a. The vigilance in ART_s was set to 1.0 so that it learned each pattern of view transitions represented in the STORE model. To make ART_a categories more general, the maximum possible ART_a vigilance was limited to $\bar{\rho}_{max} = 0.95$ during match tracking. In

this way, the architecture used sequences of length two for recognition, thereby incurring the possible cost of generating up to $O(N^2)$ sequence recognition nodes in ART_s.

In Figure 16, the black circles represent the recognition scores using view transitions for preprocessing with the large scale CORT-X 2 filters, the black squares represent recognition scores using view transitions for preprocessing with small scale CORT-X 2 filters. Comparing recognition scores from using view transitions with those using evidence accumulation or voting over two views, it can be seen that the results are similar. Since evidence accumulation does not require the temporal order information encoded by ART_s, evidence accumulation over view transitions seems preferable in the present application.

This conclusion should not be construed as implying that temporal order information is never useful in 3-D object recognition. Indeed, it is known from cognitive data that working memories which encode temporal order information occur in the brain. Whether an explicit encoding of order information, as in the Seibert–Waxman model, or an implicit encoding, as in a STORE model, are preferable remains to be seen in those situations where the tradeoff between preprocessor, categorizer, and working memory does require temporal order information for enhanced performance.

9. DISCUSSION AND GENERALIZATION TO IMAGE UNDERSTANDING

Using the smaller set of CORT-X 2 filters, a 3-D object recognition rate of approximately 90% may be achieved from single 2-D views alone without recourse to more elaborate methods of generating aspect graph models of the 3-D objects. When evidence integration or voting over an unordered sequence of views is added, recognition rates reach 98.5% within three views. Voting over two views did as well as using view transitions on this database, but without the drawback of needing to learn $O(N^2)$ view transitions given N 2-D views. In addition, it was shown that the above recognition rates can be maintained even in high noise conditions using the preprocessing methods described here.

These high recognition rates are achieved by using a different preprocessor and supervised learning to create more optimal category boundaries than in the Seibert and Waxman studies. As reported in their discussion of results in Seibert and Waxman (1992), their unsupervised clustering of coarse-coded maximal curvature data created general categories that unambiguously selected for the correct 3-D object only 25% of the time. In so doing, their network created 41 categories during training. In order to overcome the ambiguity of their general ART 2 categories, Seibert and Waxman used explicitly coded 2-D view category transitions to help identify the 3-D objects.

Using this approach, the network must be able to represent possible cross-correlations between every categorical 2-D view in the view transition matrices, one for each object, even if no correlations are eventually found between some of the categories. This is because a view transition matrix represents a definite network structure that explicitly codes the particular sequence of view transitions that ends up coding a prescribed 3-D object. Thus, such an algorithm is committed to represent all possible correlations between each node of the 41 2-D view categories. The total number of correlations is then $(41^2 - 41)/2 = 820$, since transitions and their reverse are equivalent and there are no self-transitions, this is done for each object, then a total of $820 \times 3 = 2460$ correlation matrices would be needed. Add to this the 41 ART 2 categories and up to 2501 activations could be needed to recognize the three jets. As seen in Table 6, VIEWNET needs only 35 nodes to categorize this database since, by using unordered voting, it avoids the $O(N^2)$ penalty for using view transition information given N 2-D views. If a version of VIEWNET were used that incorporates a STORE working memory, then ART sequence recognition nodes would also be needed. However, these nodes do not need to be prewired in the network. They are self-organized by learning. Only as many sequences as are actually used for prediction would need to be represented as sequence nodes in ART_s.

A number of enhancements of the VIEWNET family of architectures may be contemplated. For example, the Fuzzy ARTMAP architecture computes goodness of fit information that may be used to enhance its power in future applications. In particular, the match or choice equation, (37) or (34), respectively, in Appendix B may be used to measure the *quality* of the recognition. If VIEWNET recognizes a 3-D object, but its ART_a category prototype provides a poor fit to the input vector, then the goodness of fit information could be used to cause VIEWNET to collect more data before a final recognition decision is made. Likewise, if VIEWNET is embedded in an active vision system, then a poorly fitting view could be used to trigger the system to move its focus of attention to get a better perspective.

VIEWNET architectures may also be upgraded in several ways to handle more complex scene understanding problems. These enhancements could include:

(a) Boundary Segmentation. A more powerful boundary segmenter than a CORT-X filter, such as a BCS model (as in Grossberg et al., 1995), can

much sparser and noisier images. The BCS feedback network typically converges in a few iterations of feedback, so processing speed would not be impaired in appropriate hardware.

(b) *Surface Representation.* A system complementary to the BCS, called the Feature Contour System, or FCS, computes filled-in surface representations of a scene (Arrington, 1994; Grossberg and Todorović, 1988). These surface representations can also be used to separate figures from their backgrounds (Grossberg, 1994; Grossberg and Wyse, 1991, 1992). Fusion of boundary and surface representations can, in principle, achieve better recognition than boundary segmentation alone in cases where surface properties, such as color and relative contrast, are important. See Grossberg (1994) for a review of human psychophysical data about such data fusion.

(c) *Fusion ARTMAP.* In order to categorize data from multiple boundary and surface representations, a generalization of Fuzzy ARTMAP, called Fusion ARTMAP, can be used (Asfour, 1994, 1995; Asfour et al., 1995; Asfour et al., 1993a, b). This architecture is designed to autonomously search for that combination of input channels that correctly classifies each output prediction.

(d) *What-and-Where Filter.* In order to more effectively utilize the form information within the surface representation, a different invariant filter than the log polar filter can be used; for example, a What-and-Where filter (Carpenter et al., 1992b, 1993a). The output of log-polar-Fourier preprocessing is an invariant representation, but one that has lost information about the form of the object, as well as about the object's place in a larger scene. The What-and-Where filter is a filter-based invariant transform system in which information about the position, size, and orientation of the object is retained, and no form information is lost.

The strategy leading to this system is suggested by the brain's use of parallel streams in the visual cortex to compute *where* an object is and *what* the object is (Goodale & Milner, 1992; Mishkin et al., 1983; Ungerleider and Mishkin, 1982). The What-and-Where filter consists of a Where channel that computes the position, orientation, and size of a target figure, and a What channel that uses the information provided by the Where channel to encode an invariant object representation. Subsequent object recognition is based upon output from the What channel. The Where channel includes banks of spatial filters of varying sizes and orientations. Competition between filters yields a spatial map whose cell activations multiplex a representation of the position, orientation, and size of the figure. This information is utilized within the What channel to generate an invariant object representation. That is, the figure is transformed so that it is centered at the origin with canonical size and horizontal orientation.

(e) *Image Understanding.* The What-and-Where filter can be used to generalize VIEWNET for image understanding. The Where filter defines a spatial map whose nodes multiplex information about the position, size, and orientation of every figure in an image. In particular, activation of a node, or cell population, in this map implicitly represents all three spatial properties of the corresponding image figure. The Where filter nodes are thus distinct channels that each process at most one figure. Each channel, in turn, inputs to its own What invariant filter and recognition network. Thus the Where map of each figure is linked, or bound, to the corresponding What recognition of the figure, even though the What recognition strips the figures of its spatially variant properties. Due to this linkage, the Where spatial map and the What recognition categories can be combined into a total input vector in a more general image interpretation network (Figure 18). Such a network learns to combine information about the identities of each object with information about the objects' spatial relationships to derive a more global interpretation of scenic meaning.

Fusion ARTMAP is being designed to handle just such problems of multidimensional data fusion, classification, and prediction. In this image understanding application, Fusion ARTMAP would be used to learn those combinations of spatial and visual information that predict a desired image interpreta-

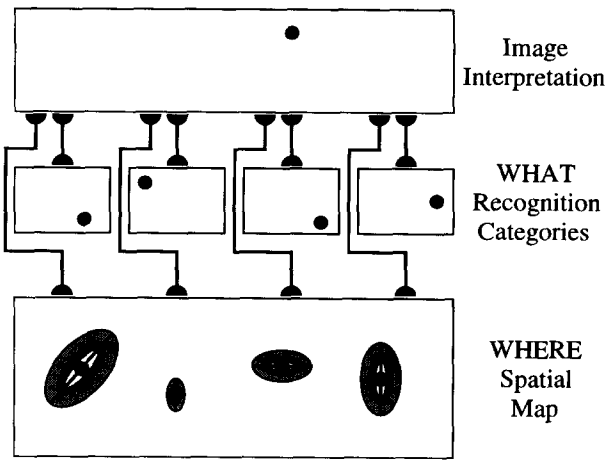

FIGURE 18. Reciprocal interactions of a Where spatial map and What recognition categories with an image interpretation network can learn scenic interpretations that combine information about multiple objects and their spatial relations. Fusion ARTMAP can be used for supervised learning of those combinations of object categories and spatial relations that reliably predict a prescribed scenic interpretation. [Reprinted with permission from Carpenter et al. (1993a).]

tion. From this broader computational perspective, VIEWNET 1 can be viewed as perhaps the simplest example of a family of self-organizing neural architectures for both 3-D object recognition and image understanding.

REFERENCES

Arrington, K. (1994). A reduction in the number of directionally selective neurons extends the spatial limit for global motion perception. *Vision Research*, **24**, 3241–3251.

Asfour, Y. (1994). Neural networks for multi-sensor fusion and classification. Ph.D. thesis, Boston, MA: Boston University.

Asfour, Y., Carpenter, G., Grossberg, S., & Lesher, G. (1993a). Fusion ARTMAP: A neural network architecture for multi-channel data fusion and classification. In *Proceedings of the world congress on neural networks* (Vol. 2, pp. 210–215). Hillsdale, NJ: Erlbaum Associates. Boston University Technical Report CAS/CNS-TR-93-05.

Asfour, Y., Carpenter, G., Grossberg, S., & Lesher, G. (1993b). Fusion ARTMAP: An adaptive fuzzy network for multi-channel classification. In *Proceedings of the third international conference on industrial fuzzy control and intelligent systems* (pp. 155–160). New York: IEEE Service Center.

Bowyer, K., Eggert, D., Stewman, J., & Stark, L. (1989). Developing the aspect graph representation for use in image understanding. In *Proceedings of the 1989 image understanding workshop* (pp. 831–849).

Bradski, G., Carpenter, G., & Grossberg, S. (1992). Working memory networks for learning temporal order with application to 3-D visual object recognition. *Neural Computation*, **4**, 270–286.

Bradski, G., Carpenter, G., & Grossberg, S. (1994). STORE working memory networks for storage and recall of arbitrary temporal sequences. *Biological Cybernetics*, **71**, 469–480.

Carpenter, G., & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**, 4919–4930.

Carpenter, G., Grossberg, S., & Mehanina, C. (1989). Invariant recognition of cluttered scenes by a self-organizing ART architecture: CORT-X boundary segmentation. *Neural Networks*, **2**, 169–181.

Carpenter, G., Grossberg, S., & Rosen, D. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, **4**, 493–504.

Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., & Rosen, D. (1992a). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, **3**, 698–713.

Carpenter, G., Grossberg, S., & Lesher, G. (1992b). A what-and-where neural network for invariant image preprocessing. In *Proceedings of the international joint conference on neural networks* (Vol. III, pp. 303–308). Piscatway, NJ: IEEE Service Center.

Carpenter, G., Grossberg, S., & Iizuka, K. (1992c). Comparative performance measures of fuzzy artmap, learned vector quantization, and back propagation for handwritten character recognition. In *Proceedings of the international joint conference on neural networks* (Vol. I, pp. 794–799). Piscataway, NJ: IEEE Service Center.

Carpenter, G., Grossberg, S., & Lesher, G. (1993a). The what-and-where filter: A spatial mapping neural network for object recognition and image understanding. Tech. report, Boston, MA: Boston University. Boston University CAS/CNS-TR-93-043.

Carpenter, G., Grossberg, S., & Reynolds, J. (1993b). Fuzzy ARTMAP, slow learning and probability estimation. *IEEE Transactions on Neural Networks*, in press. Boston, MA: Boston University. Boston University Technical Report CAS/CNS-TR-93-014.

Chang, I., & Huang, C. (1992). Aspect graph generation for non-convex polyhedra from perspective projection view. *Pattern Recognition*, **25**, 1075.

Chen, S., & Freeman, H. (1990). Computing characteristic views of quadric-surfaced solids. In *Proceedings of the 10th international conference on pattern recognition.*

Dickinson, S., Pentlend, A., & Rosenfeld, A. (1992). 3-D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**, 174–198.

Edelman, S., Bülthoff, H., & Weimshall, D. (1989). Stimulus familiarity determines recognition strategy for novel 3-D objects. Technical Report, Cambridge, MA: MIT. MIT AI Lab Memo No. 1138.

Eggert, D., & Bowyer, K. (1993). Computing the perspective projection aspect graph of solids of revolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**, 109.

Fekete, G., & Davis, L. (1984). Property spheres: A new representation for 3-D object recognition. In *Proceedings of the 1984 IEEE workshop on computer vision: representation and control*, pp. 198–201.

Freeman, H., & Chakravarty, I. (1980). The use of characteristic views in the recognition of three-dimensional objects. In E. Gelsema, & L. N. Kanal, (eds), *Pattern recognition in practice*. New York: North-Holland.

Gigus, Z., & Malik, J. (1988). Computing the aspect graph for line drawings of polyhedral objects. In *Proceedings of the IEEE Conference on Computing Vision Pattern Recognition*, pp. 654–661.

Gigus, Z., & Malik, J. (1990). Computing the aspect graph for line drawings of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Vision*, **12**, 113.

Goodale, M., & Milner, D. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, **15**, 20–25.

Grossberg, S. (1978a). Behavioural contrast in short-term memory: Serial binary memory models or parallel continuous memory models? *Journal of Mathematical Psychology*, **17**, 199–219.

Grossberg, S. (1978b). A theory of human memory: Self-organization and performance of sensory-motor codes, maps, and plans. In R. Rosen, & F. Snell, (eds), *Progress in theoretical biology* (Vol. 5, pp. 233–374). New York, NY: Academic Press.

Grossberg, S. (1987). Cortical dynamics of three-dimensional form, color, and brightness perception. I: Monocular theory. *Perception and Psychophysics*, **41**, 87–1116.

Grossberg, S. (1994). 3-D vision and figure-ground separation by visual cortex. *Perception and Psychophysics*, **55**, 48–120.

Grossberg, S., & Mingolla, E. (1985). Neural dynamics of perceptual grouping: Boundary completion, illusory figures, and neon color spreading. *Psychological Review*, **92**, 173–211.

Grosberg, S., & Todorović, D. (1988). Neural dynamics of 1-D and 2-D brightness perception: A unified model of classical and recent phenomena. *Perception and Psychophysics*, **43**, 241–277.

Grossberg, S., & Wyse, L. (1991). A neural network architecture for figure-ground separation of connected scenic figures. *Neural Networks*, **4**, 723–742.

Grossberg, S., & Wyse, L. (1992). Figure-ground separation of connected scenic figures: Boundaries, filling-in, and opponent processing. In G. A. Carpenter & S. Grossberg (eds), Neural networks for vision and image processing, pp. 161–194. Cambridge, MA: MIT Press.

Grossberg, S., Mingolla, E., & Todorović, D. (1989). A neural network architecture for preattentive vision. *IEEE Transactions on Biomedical Engineering*, **36**, 79–102.

Grossberg, S., Mingolla, E., & Williamson, J. (1995). Synthetic aperture radar processing by a multiple scale neural system for boundary and surface representation. *Neural Networks*, **8**, 1005–1028.

Koenderink, J., & van Doorn, A. (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*, **32**, 211–216.

Liu, C., & Tsai, W. H. (1990). 3-D curved object recognition from multiple 3-D camera views. *Computer Vision, Graphics, Image Processing*, **50**, 177–187.

Logothetis, N., Pauls, J., Buelthoff, H., & Poggio, T. (1994). View-dependent object recognition by monkeys. *Current Biology*, **4**, 401.

Martin, W., & Aggarwal, J. (1983). Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5**, 150–158.

Mishkin, M., Ungerleider, L., & Macko, K. (1983). Object vision and spatial vision: two cortical pathways. *Trends in Neurosciences*, **6**, 414–417.

Plantinga, H., & Dyer, C. (1990). Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, **5**, 137–160.

Poggio, T., & Edelman, S. (1990). A network that learns to recognize three-dimensional objects. *Nature*, **343**, 263–266.

Ponce, J., & Kriegman, D. (1990). Computing exact aspect graphs of curved objects: Parametric surfaces. In *Proceedings of the 8th national conference on artificial intelligence*, pp. 1074–1079.

Rieger, J. (1990). The geometry of view space opaque objects bounded by smooth surfaces. *Artificial Intelligence*, **44**, 1–40.

Rimey, R., & Brown, C. (1991). Hmms and vision: Representing structure and sequences for active vision using hidden markov models. Technical report, Rochester, NY: University of Rochester. University of Rochester Computer Science Department TR-366.

Schwartz, E. (1977). Spatial mapping in primate sensory projection: analytic structure and relevance to perception. *Biological Cybernetics*, **25**, 181–194.

Seibert, M., & Waxman, A. (1990a). Learning aspect graph representations of 3-D objects in a neural network. In *Proceedings of international joint conference on neural networks (IJCNN-90), Washington, DC* (Vol. 2, pp. 233–236).

Seibert, M., & Waxman, A. (1990b). Learning aspect graph representations from view sequences. In D. Touretzky (ed.), *Advances in neural information processing systems* (Vol. 2, pp. 258–265). San Mateo, CA: Morgan Kaufmann Publishing.

Seibert, M., & Waxman, A. (1991). Learning and recognizing 3-D objects from multiple views in a neural system. In H. Wechsler (ed.), *Neural networks for perception*. New York: Academic Press.

Seibert, M., & Waxman, A. (1992). Adaptive 3-D-object recognition from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, 107–124.

Sripradisvarakul, T., & Jain, R. (1989). Generating aspect graphs for curved objects. In *Proceedings of the IEEE workshop interpretation of 3-D scenes* (pp. 109–115).

Stewman, J., & Bowyer, K. (1990). Direct construction of the perspective projection aspect graph of convex polyhedra. *Computer Vision, Graphics, and Image Processing*, **51**, 20.

Underwood, S., & Coates, C. (1975). Visual learning from multiple views. *IEEE transactions on Computing*, **24**, 651–661.

Ungerleider, L., & Mishkin, M. (1982). Two cortical visual systems. In D. Ingle, M. Goodale, & R. Mansfield, (eds), *Analysis of visual behavior*, pp. 549–586. Cambridge, MA: MIT Press.

Winston, P. (1975). *The psychology of computer vision*. New York: McGraw-Hill.

Zadeh, L. (1965). Fuzzy sets. *Information Control*, **8**, 338–353.

Zhang, S., Sullivan, G., & Baker, K. (1992). Relational model construction and 3-D object recognition from single 2-D monochromatic images. *Image and Vision Computing*, **10**, 313.

APPENDIX A: CORT-X 2 EQUATIONS

The equations for the CORT-X 2 filter as described in Section 3 are discussed below. Figure 5(a) shows the model flow chart and Figure 5(b) shows the kernels used in the CORT-X 2 filter algorithm. Table 1 summarizes the parameters used in the simulations.

A.1. Step 1. Discounting the Illuminant

The 1D cross-sections of the on and off kernels are shown in Figure 5(a).

A.1.1. ON-C and OFF-C Network. The activation x_{ij} at node v_{ij} at position (i,j) obeys the shunting on-center off off-surround equation

$$\frac{d}{dt}x_{ij} = -Ax_{ij} + (B - x_{ij})C_{ij} - (x_{ij} + D)E_{ij}, \tag{10}$$

and $\bar{x}_{ij}$ obeys the off-center, on-surround equation

$$\frac{d}{dt}\bar{x}_{ij} = -A(\bar{x}_{ij} - S) + (\bar{B} - \bar{x}_{ij})\bar{C}_{ij} - (\bar{x}_{ij} + \bar{D})\bar{E}_{ij} \tag{11}$$

where C_{ij}, $\bar{C}_{ij}$, E_{ij}, $\bar{E}_{ij}$ are discrete convolutions of the input with Gaussian kernels of the form

$$K_{ij} = \sum_{p,q} I_{pq} K_{pqij} \tag{12}$$

with

$$K_{pqij} = \kappa \exp\{-\alpha^{-2}\log 2[(p-i)^2 + (q-j)^2]\}. \tag{13}$$

The on-center kernel of $\bar{x}_{ij}$ is the off-surround kernel of x_{ij}, and the off-surround kernel of $\bar{x}_{ij}$ is the on-center kernel of x_{ij}. Then $\bar{C}_{ij} = E_{ij}$, $\bar{E}_{ij} = C_{ij}$. Also in eqns (10) and (11), $\bar{B} = D$ and $\bar{D} = B$. At equilibrium in the ON-C network

$$x_{ij} = \frac{\sum_{(p,q)} (BC_{pqij} - DE_{pqij}) I_{pq}}{A + \sum_{(p,q)} (C_{pqij} + E_{pqij}) I_{pq}}, \tag{14}$$

and in the OFF-C network.

$$\bar{x}_{ij} = \frac{AS + \sum_{(p,q)} (DE_{pqij} - BC_{pqij}) I_{pq}}{A + \sum_{(p,q)} (C_{pqij} + E_{pqij}) I_{pq}}. \tag{15}$$

A.2. Step 2. CORT-X 2 Filter

Oriented receptive fields are elliptical as shown in Figure 5(b) so that

$$\frac{y^2}{a_s^2} + \frac{x^2}{b_s^2} = 1, \tag{16}$$

where a_s is the major axis and b_s is the minor axis with $a_s \geq b_s$. Two sizes of receptive fields were used, indexed by the subscript s with

1 = small and 2 = large scale. Orientations were chosen at angles spaced every $\pi/8$ degrees indexed below by the subscript k.

Simple Cells. The output of the pair of simple cells of scale s with activation variable $x = x_{ij}$ and receptive field orientation k is defined by

$$S_{sL}(i,j,k) = \max[L_s(x,k) - \alpha_s R_s(x,k) - \beta_s, 0] \tag{17}$$

and

$$S_{sR}(i,j,k) = \max[R_s(x,k) - \alpha_s L_s(x,k) - \beta_s, 0], \tag{18}$$

where $L_s(x,k)$ and $R_s(x,k)$ are the image inputs to the left- and right-oriented receptive fields:

$$L_s(x,k) = \frac{\sum_{(p,q)\in I_s(i,j,k)} x_{pq} w_{pq}}{\sum_{(p,q)\in I_s(i,j,k)} w_{pq}} \tag{19}$$

and

$$R_s(x,k) = \frac{\sum_{(p,q)\in r_s(i,j,k)} x_{pq} w_{pq}}{\sum_{(p,q)\in r_s(i,j,k)} w_{pq}}, \tag{20}$$

and w_{pq} is a weighting factor proportional to the area of a cell covered by the receptive field. L and R in S_{sL} and S_{sR} indicate that each receptive field is sensitive to the opposite direction-of-contrast from its companion. The ON and OFF networks have separate sets of simple cells with the ON simple cells denoted by S^+_{sL} and S^+_{sR}, and the OFF simple cells denoted by S^-_{sL} and S^-_{sR}.

Complex Cells. The complex cell output $C_s(x,k)$ is defined by

$$C_s(i,j,k) = F[S^+_{sL}(i,j,k) + S^+_{sR}(i,j,k) + S^-_{sL}(i,j,k) + S^-_{sR}(i,j,k)]. \tag{21}$$

These cells are sensitive to the spatial scale s and amount-of-contrast x with orientation k, but are insensitive to direction-of-contrast.

Hypercomplex Cells (First Competitive Stage). The hypercomplex cells $D_s(i,j,k)$ receive input from the spatial competition among the complex cells

$$D_s(i,j,k) = \max\left[\frac{C_s(i,j,k)}{\varepsilon + \mu \sum_m \sum_y C_s(p,q,m) G_s(p,q,i,j,k)} - \tau, 0\right]. \tag{22}$$

The circular oriented competition kernel $G_s(p,q,i,j,k)$ shown in Figure 5(b) is normalized such that

$$\sum_{p,q} G_s(p,q,i,j,k) = 1. \tag{23}$$

Partial cells at the kernel periphery are weighted in proportion to their area (taken to be one square unit). Grey areas in Figure 5(b) are inhibitory. Cells with centers within the one unit wide band through the middle of the kernel do not contribute to the inhibition. In our simulations, the small and large scale kernels were 2/3 the diameter of the small and large scale major axes of the elliptical filters, respectively.

Hypercomplex Cells (Second Competitive Stage). Hypercomplex cells $D_s(i,j)$ compute the competition among oriented activities $D_s(i,j,k)$ at each position. This process is simplified as a winner-take-all process

$$D_s(i,j) = D_2(i,j,K) = \max_k D_s(i,j,k), \tag{24}$$

where K denotes the orientation of the maximally activated cell.

Multiple Scale Interaction. The interaction between the small and large scales is defined by

$$B_{12}(i,j) = D_1(i,j) \sum_{p,q} D_2(p,q) U(p,q,i,j), \tag{25}$$

where the unoriented excitatory kernel $U(p,q,i,j)$ is circular as in Figure 5(b) and is normalized so that

$$\sum_{p,q} U(p,q,i,j) = 1. \tag{26}$$

In our simulations, $U(p,q,i,j)$ had a diameter 2/5 as large as the major axis of the large-scale elliptical filter. Cells covered by the kernel contribute to the excitation to the extent that their area is covered by the kernel. The smaller kernel $D_1(i,j)$ in (25) localizes boundary segments and suppresses noise near the boundary, while the larger kernel $D_2(p,q)$ suppresses noise far from the boundary.

Boundary Completion. The large detectors $D_2(i,j)$, are capable of responding at locations where pixel signal strength has been reduced by noise. Such boundary signals may, however, be poorly localized. To overcome this tradeoff between boundary completion and localization, large-scale cells interact cooperatively as

$$B_2(i,j) = D_2(i,j) \max\left[\sum_{p,q} D_2(p,q,K) O(p,q,i,j,K) - \delta, 0\right]. \tag{27}$$

Kernel $O(y,x,k)$ is defined by the one-unit-wide white strips in Figure 5(b). Cells with centers lying within the one unit wide band contribute to the cooperative process. The kernel is normalized so that

$$\sum_{(p,q) \text{ in kernel}} O(p,q,i,j,k) = 1. \tag{28}$$

In the simulations, the length of the kernel is 3/5 as long as the major axis of the large-scale ellipse.

CORT-X 2 Output. The final output of the CORT-X 2 filter is the sum of the multiple scale interaction and the cooperative process:

$$B(i,j) = B_{12}(i,j) + B_2(i,j). \tag{29}$$

APPENDIX B: FUZZY ARTMAP EQUATIONS

In the following, all architectural references are to Figure 12. Figure 19 provides a flow chart describing the operation of the architecture during the presentation of input vector **a** to Fuzzy ART module ART_a. Three parameters determine Fuzzy ART dynamics: a choice parameter $\alpha > 0$; a learning rate parameter $\beta_a \in [0,1]$; and a vigilance parameter $\rho_a \in [0,1]$.

Input Preprocessing. Input **A** into a Fuzzy ART module is normalized by preprocessing the vector **A** as

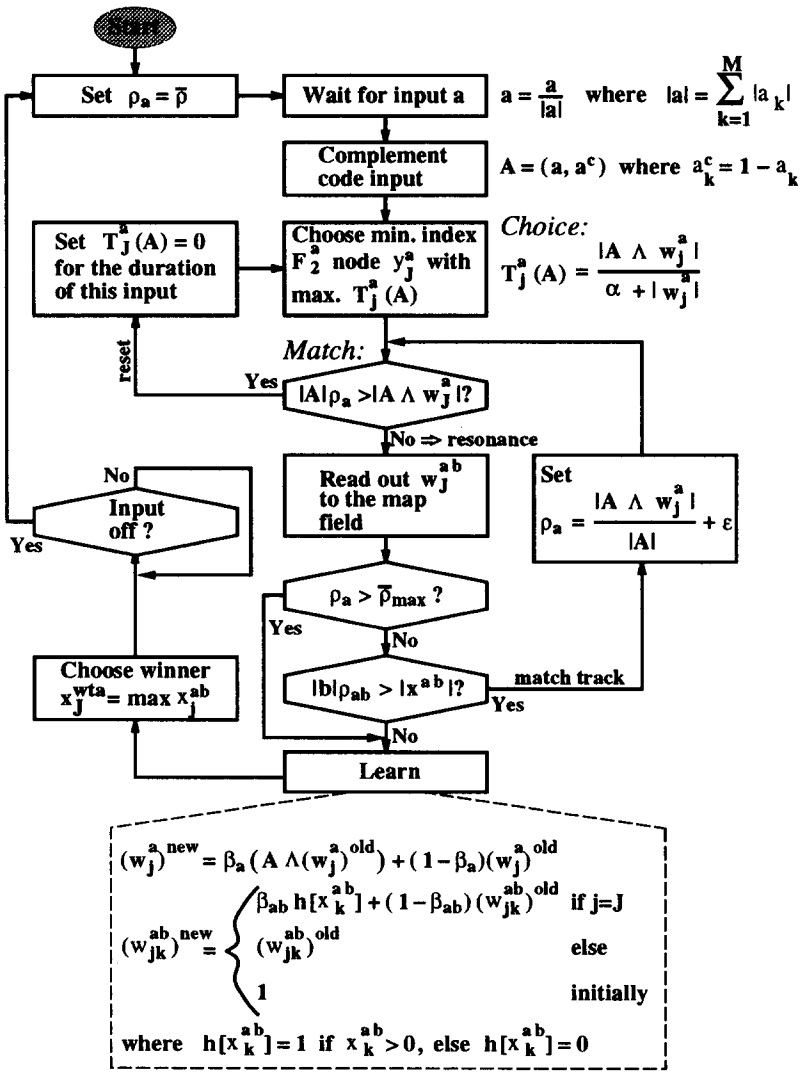

FIGURE 19. Flow-chart describing the operation of Fuzzy ARTMAP during the presentation of input vector a to Fuzzy ART module ART_a and the subsequent processing that causes the network to predict the 3-D object represented by the input 2-D view.

$$\mathbf{A} = \frac{\mathbf{a}}{|\mathbf{a}|}, \tag{30}$$

where the norm operator, $|*|$ is defined as

$$|\mathbf{a}| \equiv \sum_{k=1}^{M} |a_k|. \tag{31}$$

Inputs are then *complement coded* by setting $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ where

$$a^c_k \equiv 1 - a_k. \tag{32}$$

Category Choice. Category choice is determined by choosing the maximum choice function T^a_j,

$$T^a_J = \max\left\{T^a_j : j = 1, \ldots, N\right\} \tag{33}$$

where N is the number of nodes in F^a_2, and T^a_j is defined by

$$T^a_j(\mathbf{A}) = \frac{\left|\mathbf{A} \wedge \mathbf{w}^a_j\right|}{\alpha + \left|\mathbf{w}^a_j\right|}, \tag{34}$$

where $\mathbf{w}^a_j$ is the template belonging to the jth F^a_2 node, and the operator $\wedge$ is defined as the fuzzy AND operation

$$(p \wedge q)_k \equiv min(p_k q_k) \tag{35}$$

for M-dimensional vectors $\mathbf{p}$ and $\mathbf{q}$ (Zadeh, 1965). If more than one T^a_j is maximal, the category j with the smallest index is chosen. When category J is chosen, the node representing that category has activation $y^a_J = 1$. $y^a_j = 0$ for $j \neq J$. The F^a_1 activity vector x^a obeys

$$x^a = \begin{cases} \mathbf{A} & \text{if } F^a_2 \text{ is inactive} \\ \mathbf{A} \wedge \mathbf{w}^a_J & \text{if the } J\text{th } F^a_2 \text{ node is chosen.} \end{cases} \tag{36}$$

Match Resonance and Mismatch Reset. A match and resonance are said to occur if

$$\frac{|\mathbf{A} \wedge \mathbf{w}^a_J|}{|\mathbf{A}|} \geqslant \rho_a. \tag{37}$$

Otherwise, a mismatch reset occurs. When a reset occurs, the currently active node J is set to zero for the duration of the current input presentation and a new maximal node is found until the chosen node J satisfies eqn (37).

Map Field. The Map Field F^{ab} becomes active whenever one or both of ART_a or the supervised input b is active. Supervised inputs b_k are in one-to-one correspondence with nodes in the Map Field. If node J of F_2^a is active, then its weights $\mathbf{w}_J^{ab}$ activate F^{ab}. If a supervised input b_k becomes active, it directly activates F^{ab} node x_k^{ab} at an activation level of $b_k = 1$. The F^{ab} activity vector $\mathbf{x}^{ab}$ obeys

$$\mathbf{x}^{ab} = \begin{cases} \mathbf{b} \wedge \mathbf{w}_J^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } \mathbf{b} \text{ is active,} \\ \mathbf{w}_J^{ab} & \text{if the } J\text{th } F_2^a \text{ node is active and } \mathbf{b} \text{ is inactive,} \\ \mathbf{b} & \text{if } F_2^a \text{ is inactive and } \mathbf{b} \text{ is active,} \\ 0 & \text{if } F_2^a \text{ and } \mathbf{b} \text{ are inactive.} \end{cases} \tag{38}$$

Match Tracking. At the start of an input presentation to ART_a, the vigilance parameter is set to a baseline vigilance $\rho_a = \bar{\rho}$. Parameter ρ_{ab} is the Map Field vigilance parameter with $0 \leqslant \rho_{ab} \leqslant 1$. If

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{b}|, \tag{39}$$

then subject to

$$\rho_a \leq \rho_{max}, \tag{40}$$

match tracking causes ρ_a to be increased so that

$$\rho_a = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} + \varepsilon, \tag{41}$$

where ε is a small positive constant. If match tracking causes eqn (40) to be violated, then match tracking is inhibited and learning takes place. If after match tracking, eqn (40) is still satisfied, then a new search cycle in ART_a leads to a different F_2^a node J with

$$|\mathbf{A} \wedge \mathbf{w}_J^a| \geq \rho_a|\mathbf{A}| \text{ and } |\mathbf{b} \wedge \mathbf{w}_J^{ab}| \geqslant \rho_{ab}|\mathbf{b}|, \tag{42}$$

or, if no such node exists, ART_a is shut down until it next gets an input.

Learning. Once search ends and a winning node y_J^a is chosen in ART_a, the Fuzzy ART weight vector $\mathbf{w}_J^a$ of the winning node is updated according to

$$(\mathbf{w}_J^a)^{new} = \beta_a\left(\mathbf{A} \wedge (\mathbf{w}_J^a)^{old}\right) + (1 - \beta_a)(\mathbf{w}_J^a)^{old}, \tag{43}$$

where $\mathbf{w}_J^a(0) = 1$. The weight vectors $\mathbf{w}_j^a$, $j \neq J$ of non-winning nodes are not updated.

The weight vectors from the Fuzzy ART modules to the Map Field are updated according to

$$\left(\mathbf{w}_{jk}^{ab}\right)^{new} = \begin{cases} \beta_{ab}h[x_k^{ab}] + (1 - \beta_{ab})\left(w_{jk}^{ab}\right)^{old} & \text{if } j = J \\ (w_{jk}^{ab})^{old} & \text{if } j \neq J \\ 1 & \text{initially,} \end{cases} \tag{44}$$

where $h[x_k^{ab}] = 1$ if $x_k^{ab} > 0$, else $h[x_k^{ab}] = 0$. When $\beta_{ab} = 1$, Fuzzy ARTMAP is said to be in fast learning mode; when $0 \leqslant \beta_{ab} < 1$, Fuzzy ARTMAP is in slow learning mode.

Winner-take-all. A choice, or winner-take-all network F^{wta} sits above the Map Field F^{ab}. A winner-take-all field is needed above the Map Field when slow learning has been in effect from ART_a to the Map Field because it is then possible that a chosen F_2^a node may read out activation to more than one node in the Map Field during test mode.

The winner-take-all field is implemented algorithmically as

$$x_j^{wta} = \begin{cases} 1 & \text{if } j = \arg\max_k [x_k^{ab}], \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

Since nodes in F^{wta} are in one-to-one correspondence with nodes in the supervized field F^b, the winning node in F^{wta} represents VIEWNET's choice of the 3-D object given the single 2-D view that the network has experienced. When evidence accumulation is used, a working memory, as in Section 8, interpolates the Map Field and the winner-take-all 3-D object recognition field.