Proceedings of the IASTED International Conference
Intelligent Systems and Control 2000
August 14–16, 2000 – Honolulu, Hawaii, USA

A Locally–connected Neural Network for Fingerprint Recognition

Tyson J. Thomas
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109
tyson@brain.jpl.nasa.gov

Abstract

Fingerprint recognition is typically done by matching graphs or templates of extracted minutia from whole fingerprint images. In our approach, the local, global, and relational characteristics of an input pattern, which could be an entire fingerprint image or some sub–slice, are captured in the feature vector produced by a locally–connected neural network. This feature vector is also invariant to translations and rotations of the input pattern, transformations that are common in the registration of fingerprints. In addition, the processing elements of the neural network are particularly well suited for analog hardware implementation. This combined with the capability of using a set of consecutive fingerprint slices for recognition instead of a single large image could enable much cheaper and lower power verification systems.

Keywords: neural network, pattern recognition, fingerprint

1. Introduction

Pattern recognition systems typically consist of two stages: extraction of feature vectors[1] and classification. In the first stage, an input pattern/image comes into the system and gets reduced to a set of feature vectors through various operations (e.g. transforms, moment calculations, etc.)[2]. In the second stage, the set of feature vectors associated with the input pattern gets put into a classifier[3] that associates the feature vectors with the appropriate output class and hence the pattern is *recognized.*

Fingerprint recognition typically involves the extraction of minutia features from a whole fingerprint image, then the subsequent classification of a graph of these minutia through comparison with stored graph templates[4,5,6]. Minutia are defined as the distinguishing physical characteristics of a fingerprint such as ridge terminations and bifurcations. Both the locations and types of minutia are important to formulating a feature graph, which typically is made up of around twenty minutia points.

This work describes a novel way to extract a very useful feature vector using a locally–connected neural network, which by itself may be enough to achieve pattern recognition in certain applications if combined with a simple classifier. The technique is applied to classifying portions of fingerprint images obtained from NIST Special Database 4[7].

2. Locally–connected Neural Network[8]

Generally, the locally–connected neural network (LCNN) performs a mapping from pattern space to a lower dimensional representation space in which translational and rotational transformations in pattern space get mapped to the same place in representation space. At the same time the feature vector that results from this mapping captures local, global, and relational characteristics[9] of the input pattern, specifically as a function of the relative pixel densities. Once in representation space, a standard classification technique can be used to perform the final recognition of the feature vector.

While the locally–connected neural network algorithm can be implemented in a variety of ways, it was specifically designed to be implemented using fully parallel analog complementary metal–oxide silicon (CMOS) circuitry. Hardware implementation substantially increases processing speed while reducing power by several orders of magnitude. Analog integrated circuits for the LCNN could be combined with active pixel sensors to produce invariant pattern recognition on a single chip. For the application of fingerprint recognition explored here, however, the LCNN algorithm has been implemented with software.

2.1 Input

In hardware, the input pattern presented to the network can be stored digitally at each pixel location or can be provided directly by a sensing pixel element such as a phototransistor or other transducer. More generally, the input pattern forms the *feeding* inputs to the neuromorphs[1] spatially co–located with the input pixels.

[1] The basic functional unit of the network is called a *neuromorph* to distinguish it from generic artificial neurons or perceptrons found in the literature.

2.2 Processing

The neuromorphs use the input pattern pixels as *feeding* inputs while local neighborhood connections provide *linking* inputs, similar to the biomorphic spiking neurons of [10] which were inspired by the Eckhorn *et al.* model of the cat visual cortex neurons[11]. The entire network of neuromorphs competes for a fixed resource called *energy* that gets distributed over time across the network as a function of the relative spatial locations of *feeding* and *linking* inputs. Each neuromorph adds its feeding and linking inputs, then uses the ratio of this total to the sum of totals for all neuromorphs in the network to determine its share of the fixed energy, which becomes its new activation level. This process repeats until each neuromorph's activation level settles on a fixed value.

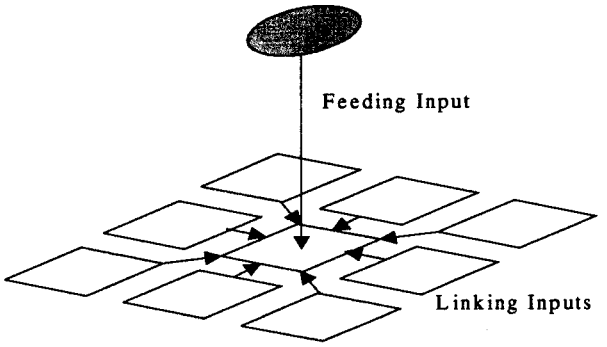

Fig. 1: Neuromorph (center square) shown with local neighborhood linking connections and pixel feeding input.

The activation of a neuromorph at any particular spatial location is thus a function of both the local *feeding* input from the corresponding input pattern pixel and the *linking* input formed from the activations at its local neighbors, as shown in Fig. 1. Even locations without any feeding input, i.e. inactive pixels, still receive influence from their neighbors and obtain a non-zero activation value. Fixing the amount of energy in the network prevents an activation explosion that could occur from the positive feedback in the linking connections. By limiting the activation resource (energy), *shunting* inhibition is introduced between all competing neuromorphs, similar to the kind of inhibition used in the eye of the Limulus crab[12].

Since the network is recurrent and therefore represented by a dynamic equation, its activation needs to be computed iteratively in computer simulations. The update for a single neuromorph goes as follows:

1. Calculate the weighted sum of the local neighborhood where the weights represent the synaptic connection or *linking* strengths –most likely all the same in order to achieve rotation and translation invariance through symmetry and uniformity.
2. Add the *feeding* input pixel value from the original input pattern to the linking input weighted sum from Step 1 to form the local sum.
3. When Steps 1 and 2 have been completed for the entire network, go back and divide the local sum by the sum of all local sums and multiply by the energy to get the new local activation level.

What happens then is that local sums are calculated for each neuromorph over the entire network and then the fraction of these sums over the total of all sums in the network is used to determine the local share of the total fixed energy. Each neuromorph therefore gets a percentage of the total energy based on what fraction its local sum was of the network total of all local sums. Mathematically, the network activation is iteratively computed by:

$$\alpha_{ij}(n+1) = \frac{I_{ij} + \sum_{kl \in N_r(ij)} [w_{ij;kl}\alpha_{kl}(n)]}{\sum_{ij}\left(I_{ij} + \sum_{kl \in N_r(ij)} [w_{ij;kl}\alpha_{kl}(n)]\right)} \cdot E$$

where,
$kl \in N_r(ij)$ are the coordinates kl of a point that falls within a radius r of the neighborhood of neuromorph ij;
$\alpha_{kl}(n)$ is the current activation level of neuromorph kl in $N_r(ij)$;
$w_{ij;kl}$ is the weight of the synaptic or linking connection between neuromorph ij and neuromorph kl;
I_{ij} is the input pattern pixel value at location ij;
E is the global network energy constant;
n is the iteration number.

2.3 Feature Vector Extraction

When the network has settled, each neuromorph has an activation level that remains fixed. A feature vector is extracted from the network by calculating an activation histogram which totals the number of neuromorphs in each of a set of different activation range bins. The number of bins in the histogram is a parameter that may be adjusted depending on the application and directly determines the dimensionality of the system's output feature vector. Generally if there are too many bins then the histogram is sensitive to slight variations in the input pattern which may be caused by edge or finite resolution effects while if there are too few bins the histogram is unable to differentiate as well between input patterns. In addition, minimum and maximum activation levels are also histogram parameters that can be determined ad hoc for the pattern type. For example, small patterns with large uniform backgrounds (e.g. written characters) will have many neuromorphs in low activation bins of the histogram that do not represent much useful information about the pattern. In this case, one might consider forming the feature vector only using bins that contain activations above a certain threshold. Conversely, for more uniform patterns (e.g. fingerprints) that have a more Gaussian looking histogram distribution, one might want to keep the full range of activations.

Every neuromorph's activation level thus falls into a bin (or alternatively a set of bins in order to make the histogram smoother and more continuous). The feature

vector is formed with the same dimension as the number of bins in the histogram, and the values in each dimension are equal to the number of neuromorphs that fall into the particular activation level's bin. To complete a classification system, a variety of techniques (nearest neighbor, neural network, etc.) may be used to perform the final classification of the above generated feature vector to complete the recognition of the input pattern.

While the process of creating a histogram from the network activations effectively destroys the specific spatial information in the original pattern, this is what produces invariance to translation and rotation (assuming symmetric weights; see Section 2.4.2). The LCNN preserves information related to the relative relationships within the pattern because local activation is a function of the relative spatial relationship between pattern pixels and their intensities.

2.4 LCNN Parameters

2.4.1 Local Neighborhood Connectivity

It does not make sense to implement a large neighborhood radius in hardware because the wiring overhead becomes substantial, reducing the number of neuromorphs that can be implemented in a single network on a chip. In software, however, it is quite trivial to expand the local neighborhood radius to any size desired, but this increases the number of calculations required per network update. The utility of higher–order neighborhood connections seems dubious for the technique described herein, so typically only first level or radius one neighborhood connections are made. Fig. 1 above showed first level neighborhood connections where the center neuromorph has linking connections to its eight nearest neighbors. Second level neighborhood connections would require an additional sixteen links to surrounding neuromorphs at radius two.

2.4.2 Weights

The weights or linking strengths constitute an additional network parameter whose influence has not yet been addressed. While the network may exhibit interesting effects as a result of non–symmetric weights, for the system described here all linking weights are equal in order to maintain the symmetry required for rotational and translational invariance. The magnitude of the linking weight, however, does change the behavior of the network without destroying the latter mentioned invariances. In practice, a larger weight multiple tends to accentuate regions of higher pattern density while blurring the network energy distribution away from the original input pattern shape. In the limit of very large weights, the input pattern is lost and the feature vector generated by the network loses its utility. Lower weights tend to preserve the structure of the original input pattern, but decrease the amount of communication between pattern regions creating a feature vector that is less representative of the relative spatial relationships between input pattern pixels. In the limit of very small weights, the feature vector generated by the network is a simple input pattern intensity histogram without any information about the relative spatial relationships of pattern regions.

3. Application: Fingerprint Recognition

The locally–connected neural network algorithm has been applied to portions of fingerprint images from the NIST Special Database 4[7] in order to generate feature vectors. These have subsequently been classified using two simple classification techniques: Euclidean Minimum Distance (EMD) and *k*–Nearest Neighbor (*k*–NN). In addition, a slightly modified version of *k*–NN is used to evaluate the performance of a system intended for verification, where the correct class is known *a priori*.

3.1 Database

The database from NIST consists of 8–bit grayscale 512 by 512 raster images of two inked impressions or "rollings" of each of 2000 different fingers. These images were produced by scanning fingerprint cards with a CCD camera. Fig. 2 shows some examples of images taken from the database.

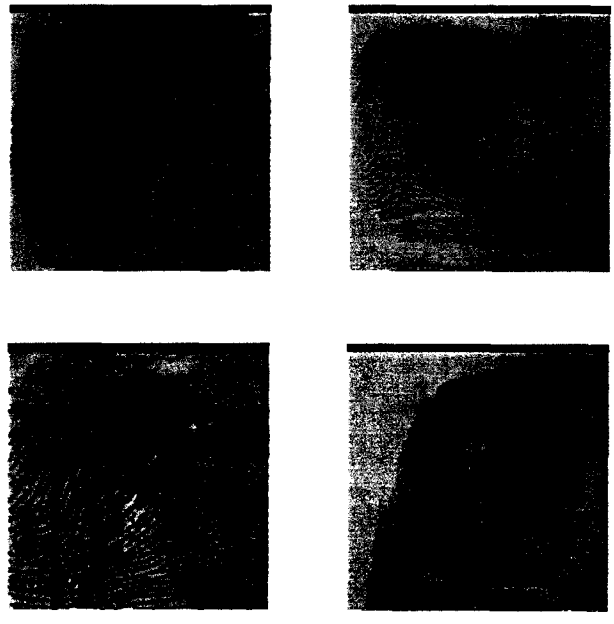

Fig. 2: Examples taken from NIST Special Database 4 showing the variety of quality and intensity in the images of ink–rolled fingerprint cards. The original images included 32 rows of white space at the bottom; these images have been shifted to the center which has produced a black bar on the top (the remaining white portion on the bottom is not visible).

Due to the background inconsistency (many images show labels or handwriting) and the variation in fingerprint sizes, 256 by 256 sub–images as shown in Fig. 3 were used for input to the LCNN. Using a smaller window has the added advantage of substantially decreasing computation time (it takes about 13 seconds to process a single sub–image on an AMD 500MHz Athlon), and forcing the classifications to be performed based on fingerprint data rather than background clutter. In addition, this validates a classification scheme based on

portions or slices of fingerprint data as might be gathered by a scanning strip imager. By identifying a sequence of slices of a fingerprint image instead of a single full image, a smaller and cheaper imager could be used while enhancing security through the compounding of the false acceptance probability. For example, if the false acceptance rate (FAR) of a fingerprint slice is 10%, then the probability of falsely accepting the first fingerprint slice in a sequence is 10%, but the probability of falsely accepting two fingerprint slices would be only 1% (and three would be 0.1%).

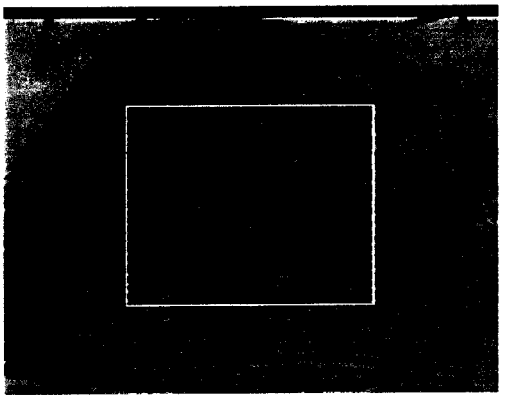

Fig. 3: A 256 by 256 sub–image of the original 512 by 512 fingerprint image. By taking the sub–image in the center, handwriting and other clutter in the background are avoided and computation time is reduced.

It would be natural to use the first–rollings for training and the second–rollings for testing in the evaluation of a classification system for these data. The consistency, however, between the separate rollings is very poor and is probably much worse quality than one would expect from a modern (non–ink based) imager. Consequently, training and testing images were taken only from the first–rolled images by sampling from the middle with a normal probability of being displaced from the center both horizontally and vertically. The standard deviation of the displacement distribution was set to 10 pixels, so that 95% of the samples could be expected to lie within ±20 from the center. This represents just under 8% relative movement in either direction, however it translates into 15% new area which brings with it new image structure since the sample is in the middle of the larger fingerprint image.

An analysis of 2000 central sub–images taken from the set of 2000 first–rollings shows an average image intensity of 102 (out of 255) with a standard deviation of 26. The minimum average intensity is 21 and the maximum is 197. This wide variance is apparent in Fig. 2 above and is also inconsistent with a modern non–inked based imager. In an effort to control this wide degree of intensity variation, which actually helps in classification by moving classes further apart, a linear stretching was applied to a set of the sub–images. This procedure rescaled image intensity to span the full grayscale range. Fig. 4 shows the results of this operation.

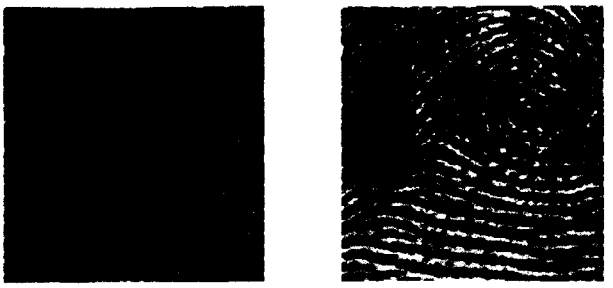

Fig. 4: The original sub–image is shown on the left and the linearly scaled version is on the right.

Gaussian noise with a standard deviation of 10 (3.9% on grayscale) was also added to each pixel of each sample to emulate sensor noise and further increase the differences between samples of the same fingerprint. Fifty samples were taken from each fingerprint used in classification to insure adequate coverage of the within–class variation.

3.2 LCNN Processing Parameters

The feature vectors generated by the LCNN from the fingerprint sub–images were obtained from empirically determined histogram parameters. Using a total energy of 100,000, a linking weight of 1, and a convergence threshold of .01 (the convergence threshold sets the point at which the network activation is considered to be settled), a histogram range of 0 to 5 with twenty–one bins was picked by analyzing the network activation statistics. This means that the feature vectors generated by the LCNN are twenty–one dimensional. For a baseline comparison of classifying effectiveness, a standard intensity histogram was also calculated and using the same number of bins as for the LCNN, twenty–one dimensional intensity feature vectors were generated.

4. Results

4.1 Euclidean Minimum Distance Classifier

The EMD classifier is very straightforward. A class mean is calculated from the training data for each fingerprint used. Test data then gets assigned to the class label of the class mean that has the smallest Euclidean distance to it. We calculated class means for 402 fingerprints using forty–nine random samples for each. The fiftieth samples were used to generate test vectors for evaluation. The number of missclassifications was calculated for the test set to generate a classification rate for a single run. Since the test vector is picked randomly from amongst the fifty samples, twenty runs were processed to get a reliable estimate for the mean classification rate for EMD processing LCNN feature vectors. We obtained a rather poor classification rate of 48.6±3.5% (the error range given is two standard deviations, or a 95% confidence interval). Using intensity feature vectors we obtained 39.4±4.4%. This would seem to indicate that the class means are not very good at representing the fingerprint classes, which seems

reasonable given that different sub–image samples contain different image structure around the edges as they are displaced about the center. While the LCNN produces translation invariant feature vectors, it produces different feature vectors when there is new structural image information present.

4.2 Nearest Neighbor Classifier

For the nearest neighbor classifier (1–NN), each training sample is considered a prototype for its class. A test vector is thus classified by the class prototype that has the minimum Euclidean distance to it. Forty–nine random samples were used to form class prototype sets for each of 402 fingerprint images, and the fiftieth sample was used as a test vector. Again, twenty runs were performed since the test vector is drawn randomly from the set of fifty with the remaining samples used as prototypes. The mean correct classification rate for LCNN feature vectors is 84.6±3.3%, while simple intensity vectors obtained 69.4±3.1%.

Another data set of 408 fingerprints, this time without the Gaussian noise added to each pixel, was similarly processed over twenty runs and obtained the excellent mean correct classification rate of 99.3±1.0%. Amazingly, the simple intensity vectors achieved 98.6±1.1%! This indicates a few things about the data set. First, despite the linear stretching applied to all the images, they are still widely varied enough to allow very good classification using simple intensity histograms. Second, the random Gaussian noise applied to pixels more strongly destroys the effectiveness of the intensity vectors, while the LCNN vectors maintain a higher level of performance since they are based on relative image structure as well as intensity. In the limit where intensity is binarized, the LCNN will still produce rich feature vectors, while the intensity vectors will be reduced to two dimensions losing almost all valuable information.

For comparison, [3] uses the Karhunen–Loeve (K–L) transform of the ridge directions for the input feature set. They first run the images through an FFT–based filter to increase the SNR in order to make the ridges more detectable. Then the local orientations of the ridges are measured at 840 equally–spaced locations. These are used as input to a translational registration module that attempts to standardize the location of the image core. Finally, the K–L transform of the modified ridge directions is taken as the feature set. Using EMD with 32 features, they obtain 73.4%, and with 1–NN they get 90.4%. These figures are not exactly comparable to those presented here for two reasons. First, while they use the same NIST Special Database 4, they perform the more difficult task of training on the first–rolled image and classifying on the second–rolled image. Second, they calculate their classification score considering that naturally occurring fingerprints have a very unequal distribution into broad class categories of Arch, Tented Arch, Left Loop, Right Loop, and Whorl, while the database has equal numbers of prints from each of these categories. One comparison, however, is fairly clear: the feature vector generated by the LCNN is much easier to calculate. This is one advantage that could enable low cost recognition in application specific hardware.

4.3 *k*–Nearest Neighbor Verifier

The *k*–NN technique uses *k* of the nearest prototypes to a test vector to form a committee, the majority of which determines class label. This gets rid of the winner–take–all of the 1–NN approach, and allows for a test vector to be classified according to the largest number of class prototypes it is near.

If we know *a priori* what class a particular test vector is supposed to be in and we are testing to see whether it is (the verification problem), then we can combine 1–NN and *k*–NN together to improve verification performance. In this scheme, if the test vector is closest to the correct class, then it is given that class assignment just like using 1–NN. If the nearest prototype is in a different class, however, we then turn to *k*–NN and determine whether the majority of *k* nearest prototypes are in the correct class. If so, then the print is correctly verified, otherwise a false rejection is given. Using the original data set with the Gaussian pixel noise back in the picture, and using *k*=49, we are able to obtain a verification rate of 88.1±2.3% using twenty runs. For simple intensity vectors the verification rate is 76.2±2.9%.

While fingerprint verification is an easier problem than recognition, it is also of more relevance to security applications and the consumer electronics market. Law enforcement is interested in quickly recognizing an individual's fingerprint from a large crime database so fingerprint data can be usable in the field. Security applications, on the other hand, are mainly concerned with verification of an individual for authorization purposes. In this scenario, the person has already represented whom they are supposed to be, and the fingerprint biometric is used to confirm this claim. The LCNN approach is more suitable to security verification applications since cost and limited computational resources are more likely to be the driving factors in these systems.

4.4 Classification Summary

	EMD	*1–NN*	*k–NN verifier*
402 prints LCNN	48.6% ±3.5%	84.6% ±3.3%	88.1% ±2.3%
402 prints Intensity histogram	39.4% ±4.4%	69.4% ±3.1%	76.2% ±2.9%
408 w/o noise. LCNN		99.3% ±1.0%	
408 w/o noise. Intensity histogram		98.6% ±1.1%	

5. Conclusions

The locally–connected neural network has been shown to produce useful feature vectors for the classification of fingerprints. On the particular database tested, simple intensity histograms also performed quite well as feature vectors for classification. It would be worthwhile to evaluate the LCNN feature vectors on data for which the simple intensity features do not perform as well in order to further validate the LCNN technique. Fingerprint data from a modern electronic or optical sensor could possibly provide such a data set. It would also be beneficial to process a larger number of fingerprints in order to insure that the representation space does not get overly crowded as additional classes are added. If the LCNN is intended to be used in a verification system, this would provide a valuable estimate of the expected false acceptance rate.

Since the LCNN was applied to fingerprint sub–images, there is further evidence that the technique could work well in a system that identifies a sequence of fingerprint image slices as opposed to a single large fingerprint image the way standard systems do. This would enable the use of smaller imagers while possibly enhancing security and decreasing the false rejection rate since the standard for any particular slice's acceptance rate could be relaxed.

The main advantage of the LCNN approach is the simplicity of the architecture and the ease of implementation with hardware. This could open up application domains that have heretofore been cost–prohibitive.

Acknowledgements

The research described herein was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology and was sponsored in part by the Ballistic Missile Defense Organization/Innovative Science and Technology Office (BMDO/IST). Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

The author would like to thank Dr. Tuan Duong, Dr. Chris Jones, Ken Hayworth, Randi Thomas, and David Weldon for useful discussions.

References

[1] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach* (London; Prentice–Hall, 1982)

[2] J. Wood, Invariant Pattern Recognition: A Review, *Pattern Recognition*, 29(1), 1996, 1–17.

[3] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson, Evaluation of Pattern Classifiers for Fingerprint and OCR Applications, *Pattern Recognition*, 27(4), 1994, 485–501.

[4] M. K. Sparrow and P. J. Sparrow, A topological approach to the matching of single fingerprints: development of algorithms for use on rolled impressions, *Technical Report Special Publication sw–124* (Washington, D.C., National Bureau of Standards, 1985)

[5] P. K. Isenor and S. A. Zapy, Fingerprint Identification Using Graph Matching, *Pattern Recognition*, 19, 1986, 113–122.

[6] A. K. Hrechak and J. A. McHugh, Automated Fingerprint Identification Using Structured Matching, *Pattern Recognition*, 23, 1990, 893–904.

[7] C. I. Watson and C. L. Wilson, *NIST Special Database 4 Fingerprint Database* (National Institute of Standards and Technology, Advanced Systems Division, Image Recognition Group; March 1992)

[8] T. J. Thomas, Locally–connected Neural Network Architecture for Invariant Pattern Recognition, *NASA Tech Briefs*, NPO–20633, to be published, 2000.

[9] R. Jain, R. Kasturi, and B. G. Schunk, *Machine Vision* (MacGraw–Hill, Inc., 1995)

[10] A. S. Baek and N. H. Farhat, Biomorphic Networks: Approach to Invariant Feature Extraction and Segmentation for ATR, *Proceedings–SPIE The International Society for Optical Engineering*, 3462 (3rd Conference), 1998, 228–241.

[11] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. Dicke, Feature Linking via Synchronization among Distributed Assemblies: Simulations of Results from Cat Visual Cortex, *Neural Computation*, 2, 1990, 293–307.

[12] H. K. Hartline and F. Ratliff, Inhibitory Interaction of Receptor Units in the Eye of *Limulus*, *Journal of General Physiology*, 40, 1957, 357–376.