Machine Learning, 15, 169-199 (1994)

Higher-Order Neural Networks Applied to 2D and 3D Object Recognition

LILLY SPIRKOVSKA (SPIRKOV@PTOLEMY.ASC.NASA.GOV)
MAX B. REID
NASA Ames Research Center, Mail Stop 269-3, Moffett Field, CA 94035-1000

Editor: Alex Waibel

Abstract. A higher-order neural network (HONN) can be designed to be invariant to geometric transformations such as scale, translation, and in-plane rotation. Invariances are built directly into the architecture of a HONN and do not need to be learned. Thus, for 2D object recognition, the network needs to be trained on just one view of each object class, not numerous scaled, translated, and rotated views. Because the 2D object recognition task is a component of the 3D object recognition task, built-in 2D invariance also decreases the size of the training set required for 3D object recognition. We present results for 2D object recognition both in simulation and within a robotic vision experiment and for 3D object recognition in simulation. We also compare our method to other approaches and show that HONNs have distinct advantages for position, scale, and rotation-invariant object recognition.

The major drawback of HONNs is that the size of the input field is limited due to the memory required for the large number of interconnections in a fully connected network. We present partial connectivity strategies and a coarse-coding technique for overcoming this limitation and increasing the input field to that required by practical object recognition problems.

Keywords. distortion invariant, pattern recognition, neural networks, higher-order, three-dimensional, two-dimensional.

1. Introduction

Robust machine vision systems are typically composed of four major tasks:

1. preprocessing, to remove noise or enhance edges,
2. segmentation, to bound objects or patterns contained in a scene,
3. object recognition, to determine the patterns present, and
4. scene understanding, to interpret the relationships between the patterns in the scene.

The emphasis of our work is on the object recognition task. Specifically, the objective of our system is to recognize an object despite changes in the object's position in the input field, its size, or its orientation, as shown in figure 1.

Traditionally, machine vision systems have separated the object recognition task into two independent subtasks: feature extraction followed by classification. The feature extraction task is concerned with finding transformations to map patterns to lower-dimensional spaces for pattern representation and to enhance class separability. Dimensionality reduction is achieved by a mapping process that projects useful information contained in the original

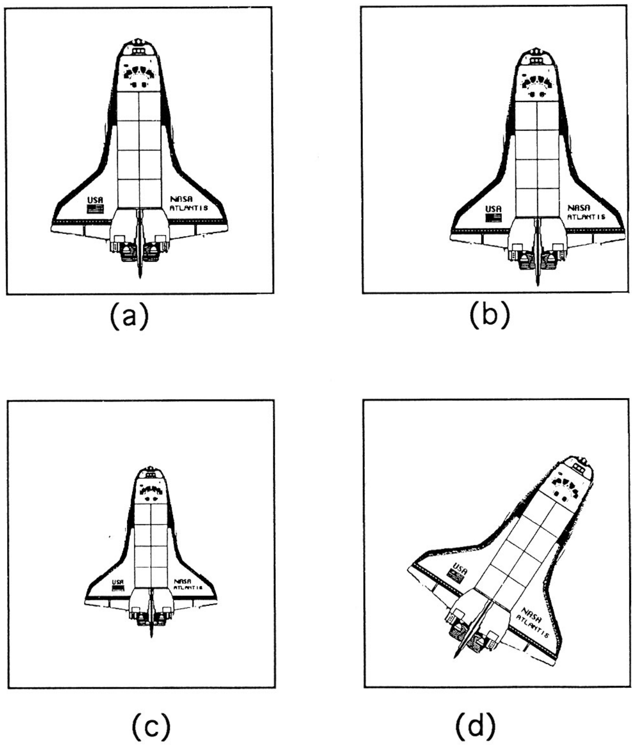


Figure 1. PSRI object recognition. In the PSRI (position, scale, and rotation invariant) object recognition domain, all four of these objects would be classified as a single object. (a) The prototype, of which three distortions are shown: (b) is a translated view, (c) is scaled, and (d) is rotated in-plane.

measurements onto a very few composite features, while ignoring redundant and irrelevant information. Examples of mappings that are invariant to one or more types of geometric transformations include moments (Hu, 1962), Fourier transforms (Haberman, 1983), chord distribution (Casasent & Chang, 1985), and circular harmonic expansions (Hsu et al., 1982).

The output of the feature extraction task, which is a vector of feature values, is then passed to the classification subtask. Based on a large set of these vectors, the classifier determines which are the distinguishing features of each object class such that new vectors are placed in the correct class within a predetermined error.

A more recent approach for position, scale, and rotation-invariant (PSRI) object recognition combines the tasks into a single system. Given only a set of views of each object class that it is required to distinguish between, the system determines which features to extract as well as which are distinguishing features of each class. The advantage of this approach is that the two subtasks can share information and improve the classifier's separating ability by extracting the useful features. The disadvantage, however, is that the system requires a longer training period, since it has no prior information about the relationship between the set of training views. Object recognition systems based on neural networks are an example of this approach.

The most popular method used in neural network-based object recognition systems is backpropagation-trained multi-layered first-order neural networks (Rumelhart et al., 1986), as illustrated in figure 2. The training process consists of applying input vectors (scaled, rotated, and translated views of each class) sequentially and adjusting the network weights using a gradient-descent learning rule until the input vectors produce the desired output vectors (class assignment) within some predetermined error. For a first-order network to learn to distinguish between a set of objects independent of their position, scale, or orientation, the network must be trained on a large set of views. The desired effect of including scaled, translated, and rotated views into the training set is that the hidden layers will extract the necessary invariant features and the network will generalize the input vectors so that it can also recognize views that are not part of the training set. Such generalization has been demonstrated in numerous simulations. However, because first-order networks do not take advantage of predefined relationships between the input nodes, they require a large number of training passes to generalize the concepts behind the transformations. Also, even after extensive training with a large training set, they usually achieve only 80%-90% recognition accuracy on novel examples (Rumelhart et al., 1986; Troxel et al., 1988).

In order to eliminate these disadvantages, higher-order neural networks (HONNs) can be used (Pitts & McCulloch, 1947; Giles & Maxwell, 1987; Giles et al., 1988; Reid et al., 1989). HONNs incorporate domain-specific knowledge into a single network. Invariance to position, scale, and in-plane rotation can be built directly into the architecture of the network and does not need to be learned. Thus, for 2D invariant object recognition, the network needs to be trained on just one view of each object, not numerous scaled, translated, and rotated views, reducing the training time significantly. To extend the invariance to the 3D object recognition domain, the network needs to be trained on a prototype view of each object and a small set of out-of-plane rotated views.

In this article, we will discuss how known relationships can be exploited and desired invariances built into the architecture of higher-order neural networks, explain the limitations of using HONNs with higher-resolution images, describe techniques to increase the

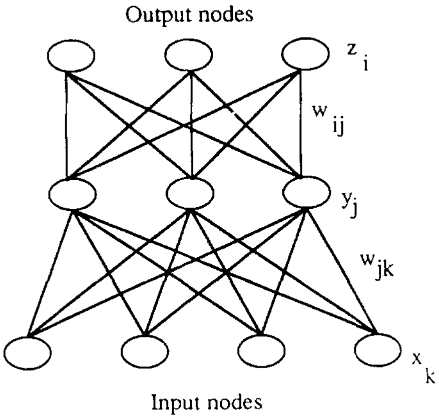


Figure 2. First-order neural network. In a first-order neural network, input nodes are connected directly to the output or hidden layer nodes. No advantage is taken of any known relationships between the input nodes.

input field size to that required by practical object recognition problems, present results for 2D object recognition both in simulation and within a robotic vision experiment and for 3D object recognition in simulation, and finally, compare our method to other approaches for position, scale, and rotation-invariant 3D object recognition.

2. Higher-order neural networks

The output of a node, denoted by y_i for node i, in a general higher-order neural network is given by

$$y_i = \Theta(\Sigma_j w_{ij} x_j + \Sigma_j \Sigma_k w_{ijk} x_j x_k + \Sigma_j \Sigma_k \Sigma_l w_{ijkl} x_j x_k x_l + \ldots) \tag{1}$$

where $\Theta(f)$ is a nonlinear threshold function such as the hard limiting transfer function given by

$$y_i = 1, \text{ if } f > 0,$$
$$y_i = 0, \text{ otherwise}; \tag{2}$$

the x's are the excitation values of the input nodes; and the interconnection matrix elements, w, determine the weight that each input is given in the summation. The interconnection weights can be constrained such that invariance to given transformations is built directly into the network architecture.

For instance, consider a second-order network as illustrated in figure 3. In a second-order network, the inputs are first combined in pairs, and then the output is determined from a weighted sum of these products. The output for a strictly second-order network is given by the function

$$y_i = \Theta(\Sigma_j \Sigma_k w_{ijk} x_j x_k). \tag{3}$$

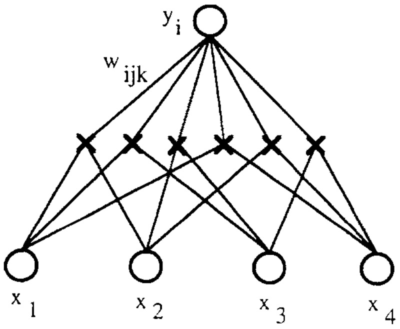


Figure 3. Second-order neural network. In a second-order neural network, the inputs are first combined in pairs (at **X**), and the output is then determined from a weighted sum of these products.

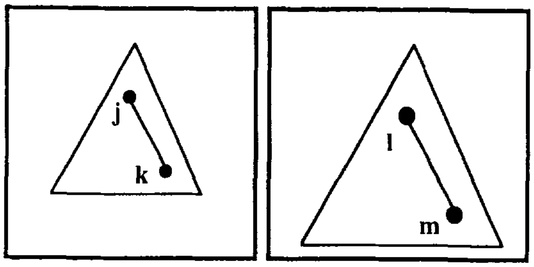


Figure 4. Translation and scale invariance in a second-order network. By constraining the network such that all pairs of points that define equal slopes use equal weights, translation and scale invariance are incorporated into a second-order neural network.

The invariances achieved using this architecture depend on the constraints placed on the weights.

As an example, each pair of input pixels combined in a second-order network define a line with a certain slope. As shown in figure 4, when an object is moved or scaled, the two points in the same relative position within the object still form the endpoints of a line with the same slope. Thus, provided that all pairs of points that define the same slope are connected to the output node using the same weight, the network will be invariant to changes in the objects' scale and position. In particular, for two pairs of pixels (j, k) and (l, m), with coordinates (x_j, y_j), (x_k, y_k), (x_l, y_l), and (x_m, y_m) respectively, the weights are constrained according to

$$w_{ijk} = w_{ilm}, \text{ if } (y_k - y_j)/(x_k - x_j) = (y_m - y_l)/(x_m - x_l). \tag{4}$$

Alternatively, the pair of points combined in a second-order network may define a distance. As shown in figure 5, when an object is moved or rotated within a plane, the distance between a pair of points in the same relative position on the object does not change. Thus, as long as all pairs of points that are separated by equal distances are connected to the output with the same weight, the network will be invariant to changes in the objects' in-plane orientation. The weights for this set of invariances are constrained according to

$$w_{ijk} = w_{ilm}, \text{ if } \|\mathbf{d}_{jk}\| = \|\mathbf{d}_{lm}\|. \tag{5}$$

That is, the magnitude of the vector defined by pixels j and k ($\mathbf{d}_{jk}$) is equal to the magnitude of the vector defined by pixels 1 and m ($\mathbf{d}_{lm}$).

To achieve invariance to translation, scale, and in-plane rotation simultaneously, a third-order network can be used.[1] As shown in figure 6, all the inputs in a third-order network are first combined in triplets and then the output is determined from the weighted sum of these products. Mathematically, this is represented by the function

$$y_i = \Theta(\Sigma_j \Sigma_k \Sigma_l w_{ijkl} x_j x_k x_l). \tag{6}$$

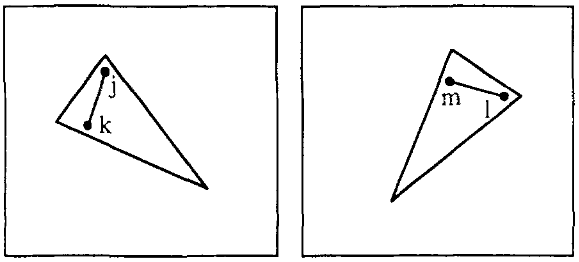


Figure 5. Translation and rotation invariance in a second-order network. By constraining the network such that all pairs of points that are equal distances away use equal weights, translation and rotation invariances are incorporated into a second-order network.

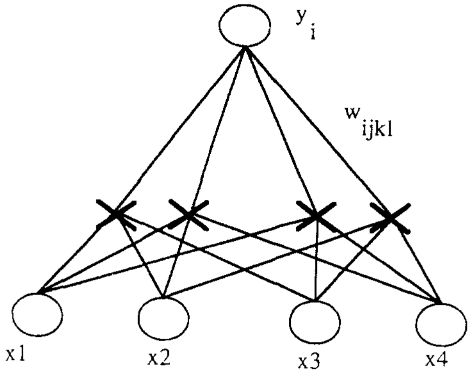


Figure 6. Third-order neural network. In a third-order neural network, input nodes are first multiplied together in triplets (at X), and then the output is determined from a weighted sum of the products.

Note that each combination of three pixels forms a triangle with some included angles (α, β, γ). Also note that triangles are invariant to changes in their position, size, or in-plane rotation, as shown in figure 7. Thus, in order to constrain the network to be invariant to position, scale, and in-plane rotation changes simultaneously, we constrain the weights such that all sets of triplets forming geometrically similar triangles are connected to the output with the same weight. That is, the weight for the triplet of inputs (j, k, l) is constrained to be a function of the associated included angles (α, β, γ) such that all elements of the alternating group on three elements (group A3) are equal:

$$w_{ijkl} = w(i, \alpha, \beta, \gamma) = w(i, \beta, \gamma, \alpha) = w(i, \gamma, \alpha, \beta). \qquad (7)$$

Note that the order of the angles matters, but not which angle is measured first.

This constraint that all similar triangles are connected to the output node with the same weight can be implemented in two steps:

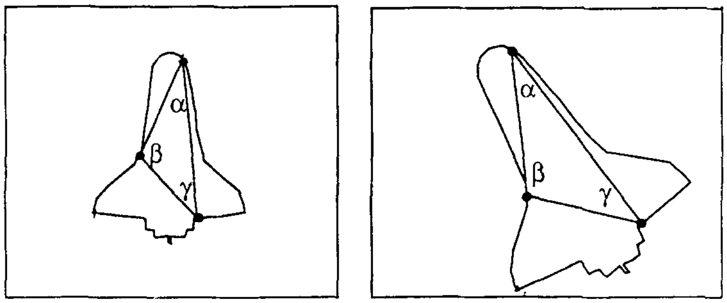


Figure 7. Position, scale, and rotation invariance in a third-order network. As long as all similar triangles are connected to the output with the same weight, a third-order network will be invariant to scale, in-plane rotation, and translation distortions.

1. Calculate the included angles α, β, and γ (to some granularity) formed by each combination of three pixels for a given input field size. Since this computation is expensive and the combination of triplets for a given input field size does not depend on the objects to be distinguished, these angles can be precalculated and stored. This step would then be modified to read the included angles corresponding to each combination of three pixels from a file, rather than calculating them.
2. Set up the correspondence between the angles α, β, and γ (same granularity) such that all triplets of the angles that are members of the alternating group (that is, the order of the angles matters, but not which one comes first) point to a single memory location. This assures that all similar triangles will manipulate the same weight value. Our implementation uses three matrices (*w*, *w_angle*, and *w_invar*) linked with pointers. Each location in *w* (indexed by the triple *i*, *j*, *k* representing the input pixels) points to a location in *w_angle* (indexed by the triple α, β, γ representing the angles formed by the triple *ijk*). Similarly, each location in *w_angle* points to a location in *w_invar*, also indexed by a triple of angles α, β, γ ordered such that the smallest angle is assigned to α. That is, *w_angle*[80][60][40] points to *w_invar*[40][80][60], as do the elements *w_angle*[60][40][80] and *w_angle*[40][80][60].

The implementation of the second-order network constraints specified by equation (4) or equation (5) requires obvious modification to the above steps.

Because HONNs are capable of providing nonlinear separation using only a single layer, once invariances are incorporated into the architecture, the weights can be modified using a simple Perceptron-like rule of the form

$$\Delta w_{ijk} = (t_i - y_i)\, x_j\, x_k, \tag{8}$$

for a second-order network, or

$$\Delta w_{ijkl} = (t_i - y_i)\, x_j\, x_k\, x_l, \tag{9}$$

for a third-order network, where the expected training output, t, the actual output, y, and the inputs, x, are all binary. We have not done a rigorous study of the computational power of these single-layer higher-order neural networks. Experimentally, we have shown that a second-order network can distinguish between 16 alphabetic characters. For third-order networks, we have only experimented with up to four objects. If the above learning rule proved insufficient to distinguish between a larger number of objects than that presented here, hidden layers could be added to the network and the learning rule changed to backward propagation of error (Rumelhart et al., 1986). Even in the more general case, the invariances would be built into the architecture of the network, and thus the multi-layer higher-order neural network would have many of the advantages presented here.

The main advantage of building invariance to geometric transformations directly into the architecture of the network is that the network is forced to treat all scaled, translated, and scaled views of an object as the same object. Invariance is achieved before any input vectors are presented to the network. Thus, the network needs to learn to distinguish between just one view of each object, not numerous transformed views.

The most severe limitation of this method is that the number of possible triplet combinations increases as the size of the input field increases. In an N × N pixel input field, combinations of three pixels can be chosen in N^2-choose-3 ways. Thus, for a 9 × 9 pixel input field, the number of possible triplet combinations is 81-choose-3 or 85,320. Increasing the resolution to 128 × 128 pixels increases the number of possible interconnections to 128^2-choose-3 or 7.3×10^{11}, a number too great to store on most machines. On our Sun 3/60 with 30 MB of swap space, we can store a maximum of 5.6 million (integer) interconnections, limiting the input field size for fully connected third-order networks to 18 × 18 pixels. To circumvent this limitation, we evaluated various strategies of connecting only a subset of input pixel triplets to the output node. In particular, we will discuss local, sampled, and regional connectivity strategies in more detail in the following section and present experimental results of each strategy's performance. The strategies were tested on the T/C recognition problem using a Sun 3/60 with 30 MB of swap space. As explained by Rumelhart et al. (1986), in the T/C problem, both objects are constructed of five squares, as illustrated in figure 8, and the problem is to discriminate between them independent of translation or 90° rotations. In our work, the network was also required to distinguish between the objects invariant to changes in scale. Thus, we used a third-order network constrained such that a selected number of all possible similar triangles are connected to the output node with the same weight. The selection process was based on the type of connectivity strategy being tested.

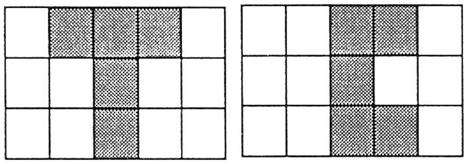


Figure 8. T/C problem. In the T/C recognition problem, each pattern consists of five squares. Over all in-plane rotations and translations, the patterns can be discriminated only if combinations of triplets of pixels are examined.

3. Partial connectivity strategies

The assumption behind *local connectivity* is that local connections are adequate for distinguishing between objects and that distant information will be represented by some combination of local information. Thus, a connection for a triplet of pixels is formed only if all three inter-pixel distances are less than some predefined bound, as shown in figure 9. Figure 10 shows the probability of connecting a given pixel triplet to the output node as a function of the greatest distance between any two of the three pixels. For local connectivity, the probability of connection is 1 for triplets of input pixels that are within a bounded distance away from each other and 0 for more distant pixel triplets. In our experiments, a third-order network using only local connections was able to distinguish between a T and a C with 100% accuracy only if pixels at least one third of the width of the input field away from each other were included.[2] Considering this distance limitation, a field of 24 × 24 pixels can be represented within the available memory, increasing the effective size by approximately 75%.

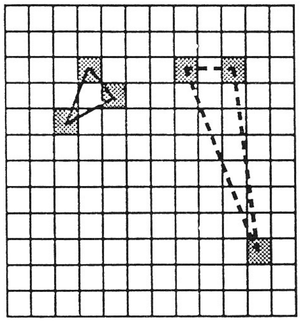


Figure 9. Local connectivity. Using local connectivity, only triplets of pixels that are close together are connected to the output (solid line). Triangles formed by using distant pixels (dashed line) are ignored.

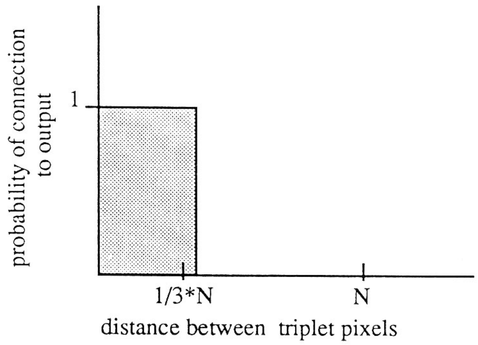


Figure 10. Probability graph for local connectivity. In local connectivity, only connections between pixels in gray are formed. N is the input field width.

In *sampled connectivity*, only a fraction of all possible input pixel triplets are connected to the output. This fraction is not a function of the distance between the pixels, as shown in figure 11. Instead, only a predetermined subset of interconnections (for example, every *i*th one) is used. The expectation is that the images will be sufficiently sampled to allow discrimination. In practice, however, this strategy fails to achieve 100% accuracy because scaled, translated, or in-plane rotated views of the same image may not be well represented by the same sampled connections. For example, assume only the interconnections in figure 12a are used for recognition. In the images in figure 12b, a T is represented by triangle 1, whereas a C is represented by triangle 2. When the images are translated and rotated as in figure 12c, triangle 1 is now associated with a C and triangle 2 with a T. Consequently, accurate recognition is not achieved. In fact, 100% accuracy can be guaranteed using this type of technique only if the fraction of interconnections used is equal to 1. Thus, this technique is not viable as a means of reducing the number of interconnections while maintaining the accuracy of performance required.

The third approach we studied is *regional connectivity*, in which triplets of pixels are connected to the output only if the distance between all the pixels falls within a *set* of preselected regions, as shown in figure 13. Regions are chosen so as to sample the space sufficiently for accurate discrimination. Thus, triplets are formed from pixels that are local, distant, and in-between. Further, because the connections are deterministic, it is guaranteed that the same information will be sampled from a rotated or a translated test image as from the training image. Additionally, since pixel triplets of various separations are sampled, some amount of scale invariance is expected.

Using regional connectivity, we were able to increase the input image resolution for the T/C problem to 64 × 64 pixels while still achieving in-plane 90° rotation and translation invariant recognition with 100% accuracy. The 64 × 64 pixel third-order network used only 5.5 million interconnections[3] instead of the fully interconnected network, which requires approximately 10 billion. This represents a reduction of the required memory by a factor of approximately 2000.

Using the same network and connectivity regions as for the T/C problem, the system also learned to distinguish between practical images such as a Space Shuttle Orbiter versus an F-18 aircraft (figure 14) in a 64 × 64 pixel input field. In this case, training took just

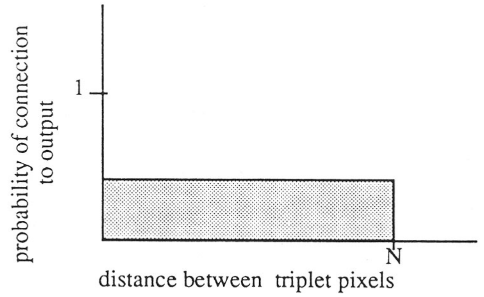


Figure 11. Sampled connectivity. In sampled connectivity, a fraction of all possible input triplets are connected, independent of the distance between them.

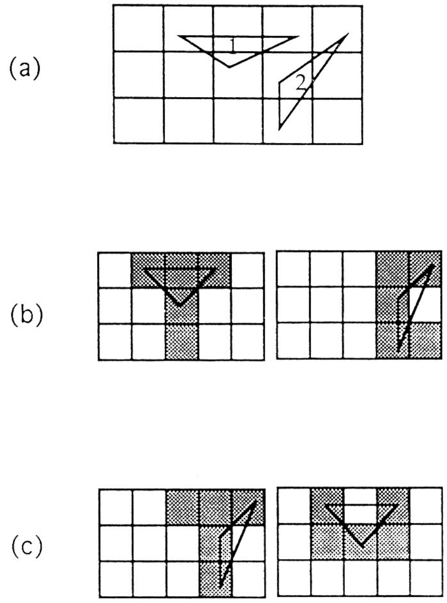


Figure 12. Problems with sample connectivity. (a) In a sampled connectivity system, only a predetermined, limited number of interconnections are formed. (b) If the above T and C are used for training, triangle 1 in (a) is associated with the T, whereas triangle 2 is associated with the C. (c) If the images in (b) are distorted, recognition fails because triangle 1 is now on for a C and triangle 2 is on for a T.

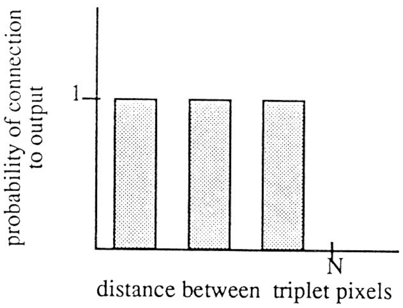


Figure 13. Regional connectivity. Only connections between pixels in gray are used in a regional connectivity system.

35 passes through the training set, which consisted of just one view of each aircraft. As for the T/C problem, the network achieved 100% recognition accuracy of translated and in-plane 90° rotated views of the two images. Furthermore, because the training objects did not completely fill the input field, we were able to verify the performance of third-order networks with non-90° rotations. In particular, the network recognized 30° and 60° rotated views, provided the objects were drawn without introducing pixellation noise.

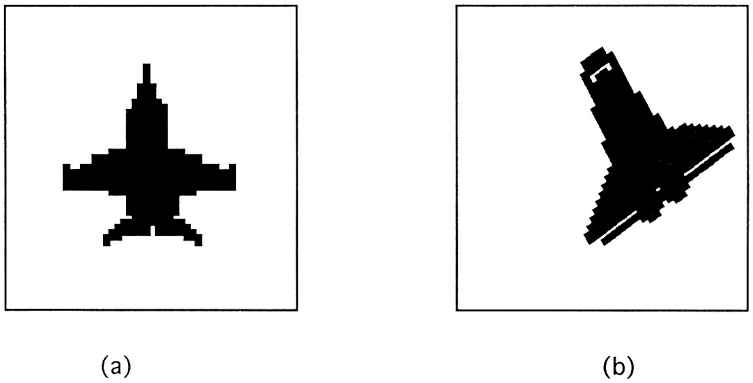


Figure 14. Sample training pair. A 64 × 64 pixel binary image of (a) an F-18 aircraft and (b) a Space Shuttle Orbiter.

In summary, by using regional connectivity we increased the input field size of HONNs to 64 × 64 pixels while still retaining many of their advantages, such as a small number of training passes, training on only one view of each object, and successful recognition invariant to in-plane rotation and translation. However, we were unable to recognize images invariant to changes in scale.

One of the problems with attaining scale invariance by connecting all similar triangles to the output node is due to the fact that when an object is scaled, many new triangles can be introduced. For example, when the object in figure 15a is scaled as shown in figure 15c, not only are the triangles in figure 15b present but also another 556 new triangles are introduced, two of which are shown in figure 15c. As explained by Spirkovska and Reid (1992), the introduction of non-similar triangles can lead to incorrect classification of scaled views of an object. In order to decrease the number of new triangles formed, we modified the preprocessing task to produce edge-only images and achieved very limited (1%-2%) scale invariance.

Another major problem of regional connectivity is processing speed. As the input field size increased, the amount of time for each pass on a sequential machine increased dramatically. The 64 × 64 pixel input field network required on the order of days on a Sun 3/60 to learn to distinguish between two objects, despite the fact that the number of interconnections was greatly reduced from the fully connected version. The number of logical comparisons required to determine whether the distances between pixels fall within the preselected reigons is prohibitively large.

In the following section, we will describe a coarse-coding algorithm that solves the problems encountered with regional connectivity, namely, the inability to increase the input field size to more than 64 × 64 pixels, lack of scale invariance, and slow processing speed. Coarse coding allows a third-order network to be used with an input field size practical for object recognition problems while still retaining its ability to recognize images that have been scaled, translated, or rotated in-plane in an input field of at least 4096 × 4096 pixels. Training takes just a few passes, and training time is on the order of minutes on a Sun 3/60, instead of days for regionally connected networks. On a Sun SPARCstation 2, training time is less than one second.

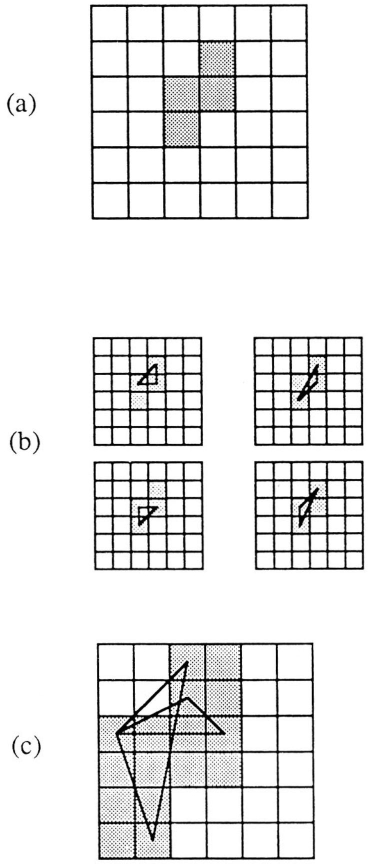


Figure 15. Illustration of the problem associated with scale invariance. (a) A sample object; (b) all the possible triangles that can be formed within the object in (a); and (c) a scaled view of the object in (a). Two examples of the 556 new triangles introduced by scaling are shown.

4. Coarse coding

The coarse-coding representation we use involves overlaying fields of coarser pixels in order to represent an input field composed of smaller pixels (Rosenfield & Touretzky, 1988; Sullins, 1985), as shown in figure 16. Figure 16a shows an input field of size 10 × 10 pixels. In figure 16b, we show two offset but overlapping fields, each of size 5 × 5 "coarse" pixels. In this case, each coarse field is composed of pixels that are twice as large (in both dimensions) as in figure 16a. To reference an input pixel using the two coarse fields requires two sets of coordinates. For instance, pixel (x = 7, y = 6) on the original image would be referenced as the set of coarse pixels ((x = D, y = C) & (x = III, y = III)), assuming a coordinate system of (A, B, C, D, E) for coarse field 1 and (I, II, III, IV, V) for coarse field 2. In order to represent pixels that are not intersected by all coarse fields, such as

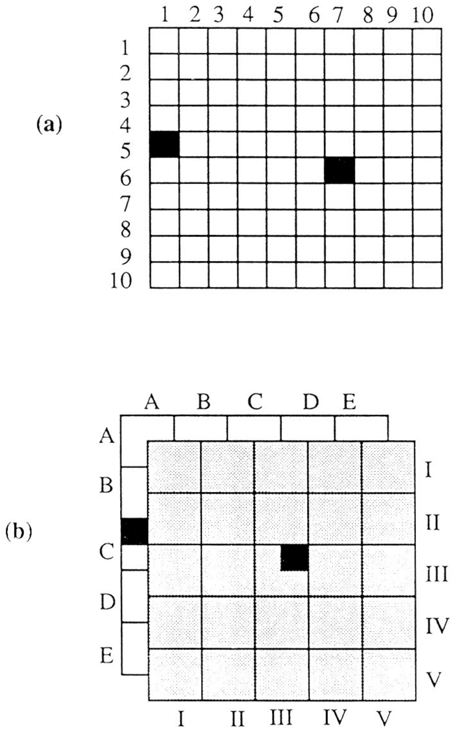


Figure 16. Coarse coding. (a) An example of a 10 × 10 pixel input field, (b) coarse-coded as two offset but overlapping fields, each of size 5 × 5 "coarse" pixels. Each coarse field is composed of pixels that are twice as large (in both dimensions) as in (a).

pixel (1, 5), coarse coding can be implemented either by using wraparound or by using only the intersection of the fields. If coarse coding is implemented using wraparound, pixel (1, 5) could be represented as the set of coarse pixels ((A, C) & (V, II)). On the other hand, if coarse coding is implemented as the intersection of the coarser fields, the two fields shown in figure 16b would be able to uniquely describe an input field of 9 × 9 pixels, not 10 × 10.

Using wraparound, the relationship between the number of coarse fields (n), input field size (IFS), and coarse-field size (CFS) in each dimension is given by

$$\mathrm{IFS} = (\mathrm{CFS} * n). \tag{10}$$

On the other hand, using the intersection-of-fields implementation, the relationship between number of coarse fields, input field size, and coarse-field size in each dimension is given by

$$\mathrm{IFS} = (\mathrm{CFS} * n) - (n - 1). \tag{11}$$

The effective input field size, IFS, is not significantly different with either implementation for small n. Further, either implementation yields a one-to-one transformation. That is, each pixel on the original image can be represented by a unique set of coarse pixels.

The above transformation of an image to a set of smaller images can be used to greatly increase the resolution possible in a higher-order neural network. For example, a fully connected third-order network for a 100 × 100 pixel input field requires 100^2-choose-3 or 1.6×10^{11} interconnections. Using 10 fields of 10 × 10 coarse pixels requires just 10^2-choose-3 or 161,700 interconnections, accessed once for each field. The number of required interconnections is reduced by a factor of about 100,000.

As an example of how coarse coding can be applied to HONNs, consider training the network to distinguish between a T and a C in an 8 × 8 pixel input field, as shown in figure 17. With coarse coding implemented with wraparound, as explained previously, there are two possible combinations that will provide an effective input field of 8 × 8 pixels: two fields of 4 × 4 coarse pixels or four fields of 2 × 2 pixels. Both possibilties are shown in figure 17b.

Applying coarse coding by using two fields of 4 × 4 pixels, the two images shown in figure 17a are transformed into the four images shown in figure 17c. Training of the network then proceeds in the usual way, with one modification: the transfer function thresholds the value obtained from summing the weighted triangles over *all* coarse field representations of each training object. That is,

$$y = 1, \text{ if } \{\Sigma_n(\Sigma_j\ \Sigma_k\ \Sigma_l\ w_{jkl}\ x_j\ x_k\ x_l)\} > 0,$$

$$y = 0, \text{ otherwise,} \qquad (12)$$

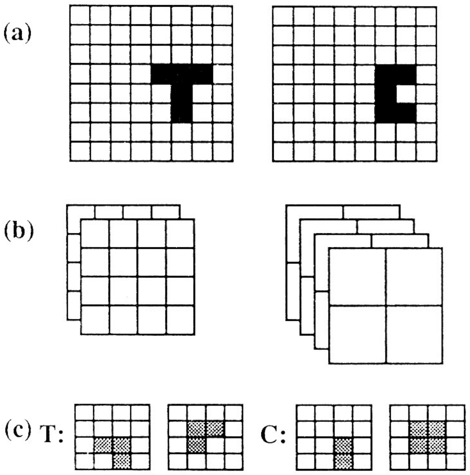


Figure 17. Using coarse-coded fields with higher-order neural networks. (a) Two training images in an 8 × 8 pixel input field; (b) two possible configurations of coarse pixels to represent the input field in (a); and (c) coarse-coded representation of the training images in (a) using two layers of 4 × 4 coarse pixels.

where j, k, and l range from one to the coarse pixel size squared, n ranges from one to the number of coarse fields, the x's represent coarse pixel values, and w_{jkl} represents the weight associated with the triplet of inputs (j, k, l).

A more detailed analysis of coarse coding and its applicability with respect to higher-order neural networks is presented by Spirkovska & Reid (1992). Here we present just a brief synopsis of its limitations, including the minimum and maximum coarse field size, the minimum and maximum number of fields that can be used and still achieve position, scale, and rotation-invariant recognition, and the maximum input field resolution possible.

The minimum possible coarse-field size is determined by the training images. The network is unable to distinguish between the training images when the size of each coarse pixel is increased to the point where the training images no longer produce unique coarse-coded representations. As an example, for the T/C recognition problem, the minimum coarse-field size that still produces unique representations is 3 × 3 pixels.

In contrast, the maximum limit is determined by the HONN architecture and the memory available for its implementation, and not by the coarse-coding technique itself. As discussed previously, the number of possible triplet combinations in a third-order network is N^2-choose-3 for an $N \times N$ pixel input field, and given the memory constraints of our Sun 3/60, the maximum possible coarse-field is 18 × 18 pixels.

Regarding the number of coarse fields that can be used and still achieve PSRI object recognition, the minimum is one field, whereas the maximum has not yet been reached. A minimum of one coarse field represents the non-coarse-coded HONN case. In order to determine the limit for the maximum number of fields possible, we ran simulations on the T/C problem coded with a variable number of 3 × 3 coarse pixels. A third-order network was able to learn to distinguish between the two characters in less than 10 passes in an input field size of up to 4095 × 4095 pixels using 2047 fields.[4] Increasing the number of fields beyond this was not attempted, because 4096 × 4096 is the maximum resolution available on most image-processing hardware that would be used in a complete HONN-based vision system. Also, each object in such a large field requires 16 MB of storage space. It takes only a few such objects to fill up a disk.[5]

Finally, as with the maximum number of coarse fileds, the maximum input field resolution possible with coarse-coded HONNs has not been delimited. As discussed above, we trained a third-order network on the T/C problem in up to a 4096× 4096 pixel input field. We expect that a resolution of 4096 × 4096 is sufficient for most object recognition tasks. Notwithstanding, we also expect that a greater resolution is possible.

5. 2D object recognition

5.1. Noise-free test images

We evaluated the performance of HONNs on numerous object recognition problems, including an SR-71/U-2 discrimination problem and an SR-71/Space Shuttle discrimination problem. The simulations used a coarse-coded third-order network designed for a 127 × 127 pixel input field. For ease of implementation, we used the intersection-of-fields approach to coarse coding with nine fields of 15 × 15 coarse pixels. The training sets were generated from actual models of the aircraft. The 8-bit gray-level images of the aircraft are shown in figure 18. The images were thresholded to produce binary images, and then edge detected using a digital Laplacian convolution filter with a positive derivative to produce the images shown in figure 19. For rotated and scaled views of the objects, the original gray-level images were first scaled, then rotated, and then thresholded and edge-detected. Notice that the profiles of the SR-71 and Space Shuttle are somewhat similar, whereas those of the SR-71 and U-2 are very different.

For both recognition problems, the network learned to distinguish between the aircraft in less than 10 passes through the training set consisting of just one view of each object.[6] The networks were then tested on a set of rotated, scaled, and translated views. A complete test set of translated, scaled, and 1° rotated views of the two objects in a 127 × 127 pixel input field consists of over 100 million images. Assuming a test rate of 200 images per

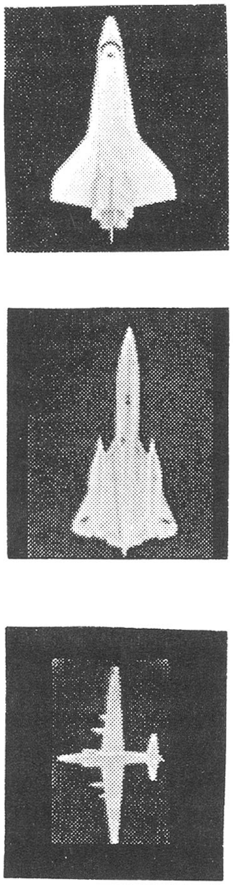


Figure 18. Eight-bit gray-level images from which the training images are generated: (a) Space Shuttle, (b) SR-71, and (c) U-2.

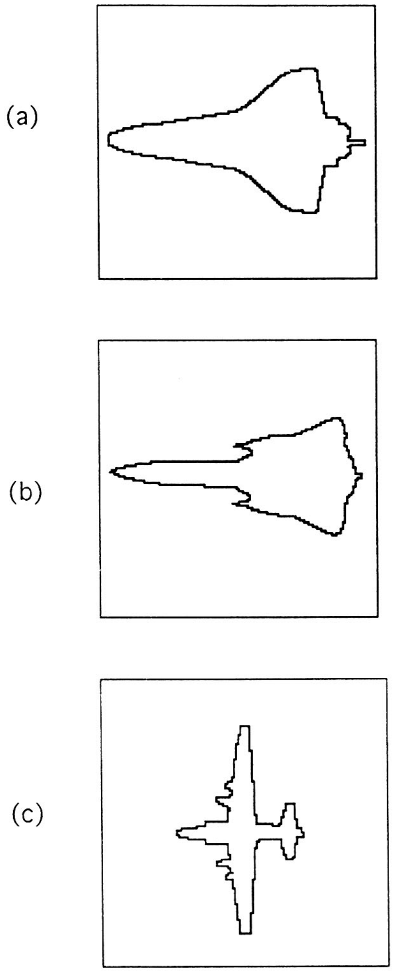


Figure 19. Training images. One binary edge-only view each for (a) Space Shuttle Orbiter vs. (b) SR-71 vs. (c) U-2.

hour, it would take computer-years to test all possible views. Accordingly, we limited the testing to a representative subset consisting of four sets:

1. All translated views, but with the same orientation and scale as the training images
2. All views rotated in-plane at 1° intervals, centered approximately at the same position and of the same size as the training images
3. All scaled views of the objects, in the same orientation and centered at the same position as the training images
4. A representative subset of approximately 100 simultaneously translated, rotated, and scaled views of the two objects.

For both recognition problems, the networks achieved 100% accuracy on all test images in sets 1 and 2. Furthermore, the networks recognized, with 100% accuracy, all scaled views from test set 3, down to 70% of the original size. Finally, for test set 4, the network correctly recognized all images larger than 70% of the original size, regardless of the orientation or position of the test image.

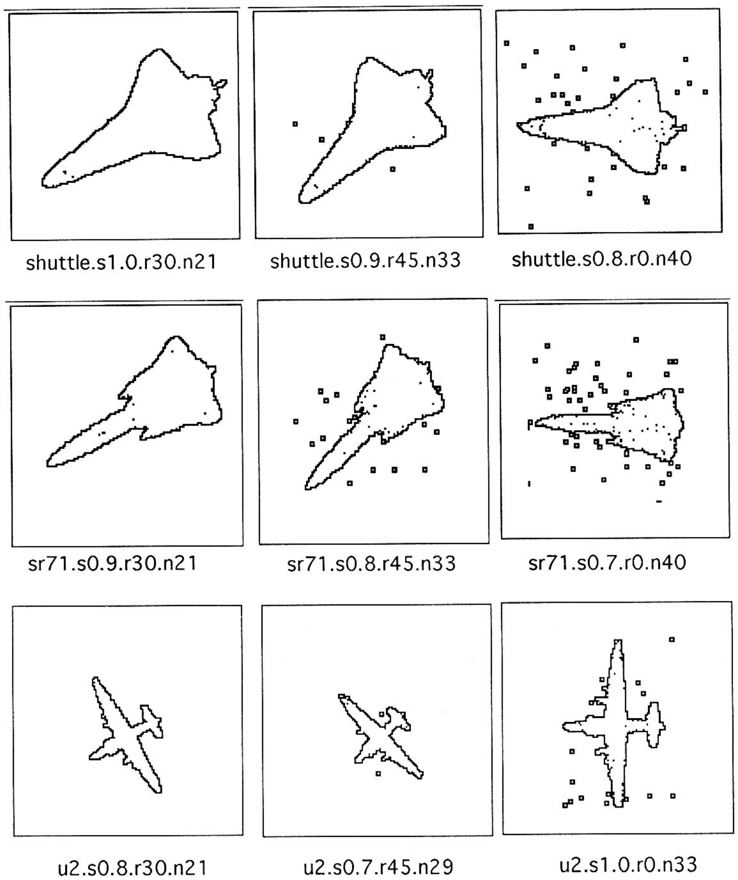
As shown in previous research (Reid et al., 1989; Spirkovska & Reid, 1992), invariance to scale is affected by the resolution to which the angles α, β, and γ in equation (7) are calculated. Briefly, as the resolution of the input field is increased, the resolution to which α, β, and γ are calculated can also be increased, generally increasing scale invariance. Scale invariance also varies with the coarse-field size and the number of coarse fields used. In general, a larger coarse-field size yields greater scale invariance. However, the learning time also increases as the coarse-field size increases. Thus, if less scale can be tolerated, a desired input-field size can be represented with a smaller coarse-field size and greater number of coarse fields.

5.2. *Noisy test images*

We also evaluated the performance of HONNs with noisy test images on the same two object recognition problems discussed above: the SR-71/U-2 discrimination problem and the SR-71/Space Shuttle discrimination problem. In particular, following training on just one view of each object, the networks were tested on numerous non-ideal images in order to determine the performance of higher-order neural networks to white Gaussian noise and partial occlusion.

To test the tolerance of higher-order neural networks to white noise, each instantiation of the network (one for the SR-71/U-2 problem and one for the Shuttle/SR-71 problem) was tested on 1200 images generated by modifying the 8-bit gray-level values of the original images using a Gaussian distribution of random numbers with a mean of 0 and a standard deviation between 1 and 50. The test set consisted of 50 images (with increasing standard deviation from 1 to 50)[7] for each of the four scales ranging from 70% to 100% in 10% increments, and each rotation of 0°, 30°, and 45°. For 90° angles, our rotation routine produces an image in which all combinations of three input pixels produce the same included angles as those produced by the unrotated image. Thus, the three angles used represent a much wider variety of rotated images than is initially apparent. The noisy images are then binarized and edge-detected. Typical test images, along with their values for standard deviation (σ), scale, and rotation, are shown in figure 20.

The results are summarized in figure 21. The network performed with 100% accuracy for our test set for a standard of up to 23 on the SR-71/U-2 problem and 26 on the Shuttle/SR-71 problem. For the similar images of the Shuttle and SR-71, the recognition accuracy quickly decreased to 75% at a σ of 30 and to 50% (which corresponds to no better than random guessing) for σ greater than 33. The SR-71/U-2 remained above 75% accuracy up to a σ of 35 (or about 14% of the gray-level range) and gradually decreased to 50% at a σ of 40 (or about 16% of the gray-level range). If we define "good performance" as greater than 75% accuracy, HONNs have good performance for σ up to 35 (or about 14% of the gray-level range) for images with very distinct profiles and σ up to 30 (or about 12% of



shuttle.s1.0.r30.n21 shuttle.s0.9.r45.n33 shuttle.s0.8.r0.n40

sr71.s0.9.r30.n21 sr71.s0.8.r45.n33 sr71.s0.7.r0.n40

u2.s0.8.r30.n21 u2.s0.7.r45.n29 u2.s1.0.r0.n33

Figure 20. Typical images used to test the tolerance of HONNs to white Gaussian noise. Test images were generated automatically by adding a normallly distributed random gray value (with a mean of 0 and a standard deviation from 1 to 50) to the original gray-level value and then binarizing and edge-detecting the resulting image. Values for scale, rotation, and standard deviation are shown under each image. For example, "s0.7.r30.n20" is an image of scale 70%, rotation 30°, and standard deviation of 20. For each object, we show a noise level recognized with an accuracy of 100%, 75%, and 50%.

the gray-level range) for images with similar profiles. It should be noted that these results apply only to images with an ideal separation of background/foreground gray levels. For images with a lower contrast, the performance may be quite different.

To test the tolerance of HONNs to partial occlusion, the two instantiations (one for the Shuttle/SR-71 problem and one for the SR-71/U-2 problem) of the third-order network built to be invariant to scale, in-plane rotation, and translation as described above were tested on occluded versions of the image pairs for each of four scales ranging from 70% to 100% in 10% increments, and each rotation of 0°, 30°, and 45°. Again, as for white noise, these three angles represent a much wider variety of rotated images than is initially apparent. We started with binary, edge-only images and added automatically generated occlusions

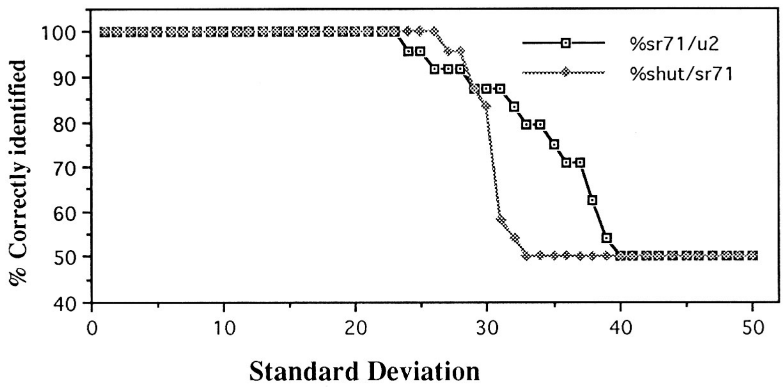


Figure 21. Tolerance of HONNs to white Gaussian noise introduced as in figure 20.

based on four variable parameters: the size of the occlusion, the number of occlusions, the type of occlusion, and the position of the occlusion. Objects used for occlusion were squares with a linear dimension between 1 and 29 pixels. The number of occlusion objects per image varied from one to four, and the randomly chosen type of occlusion determined whether the occlusion objects were added to or subtracted from the original image. Finally, the occlusions were randomly (uniform distribution) placed on the profile of the training images. The test set consisted of 10 samples for each combination of scale, rotation angle, occlusion size, and number of occlusions for a total of 13,920 test images per training image or 27,840 test images per recognition problem. Typical test images are shown in figure 22.

As shown in figure 23,, the performance of HONNs with occluded test images depends mostly on the number and size of occluding objects and to a lesser degree on the similarity of the training images. In the case of the Shuttle/SR-71 recognition problem, the network performed with 100% accuracy for our test set of one 16-pixel occlusion and up to four 10-pixel occlusions. The network performed with better than 75% accuracy ("good performance") for up to four 19-pixel occlusions, three 21-pixel occlusions, two 24-pixel occlusions, and one 29-pixel occlusion.

For the SR-71/U-2 problem, the network exhibited good performance for the entire test set, but achieved 100% accuracy only for one 4-pixel occlusion and up to four 3-pixel occlusions.

5.3. Laboratory-generated test images

At the NASA Ames Research Center, we developed a system to test algorithms for autonomous construction, inspection, and maintenance of space based habitats. The primary task of the system is to identify and grasp an arbitrary tool moving in space with all six degrees of freedom without using any kind of cooperative marking techniques. This is representative of one task required from the Flight Telerobotic Servicer (FTS) or the EVA Retriever, both of which are robots designed to operate in a weightless environment.

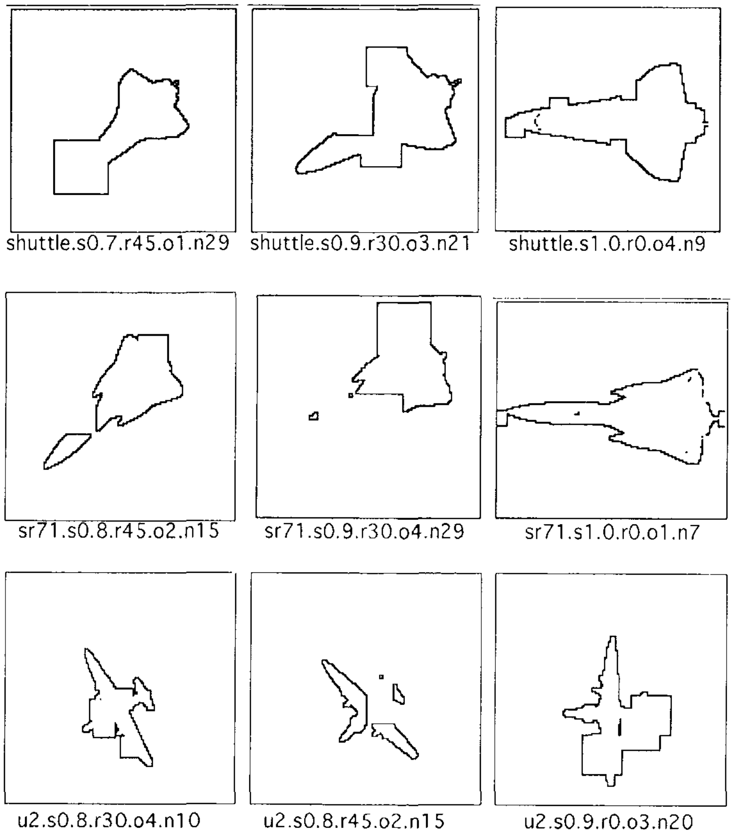


Figure 22. Typical images used to test the tolerance of HONNs with respect to occlusion.

Our test system, shown in figure 24, is a scaled-down version of the above system. It is composed of a Microbot robotic arm with a camera mounted so as to observe the workspace below its, a work surface partitioned into four distinct areas (thus eliminating the need for segmentation), and four objects. The work area is lined with black cloth to control the amount of background clutter. The training phase used ideal lighting conditions (provided by a lamp placed such that it illuminated the object evenly) so that objects' distinct features were conspicuous. However, the testing phase used only the room lighting, thus introducing errors that may be encountered in an authentic application. To test invariance to translation and in-plane rotation, the operator could arrange the objects in any order, at any position within the partition, and at any orientation. To test scale invariance, the operator could change the height parameter of the robot arm, thus controlling the distance of the object from the camera. And finally, to test tolerance to occlusion, the object could be placed partially outside its allotted partition.

We tested HONNs as the object recognition part of the above system. The HONN was trained on the four objects shown in figure 25. After 97 passes, the network identified

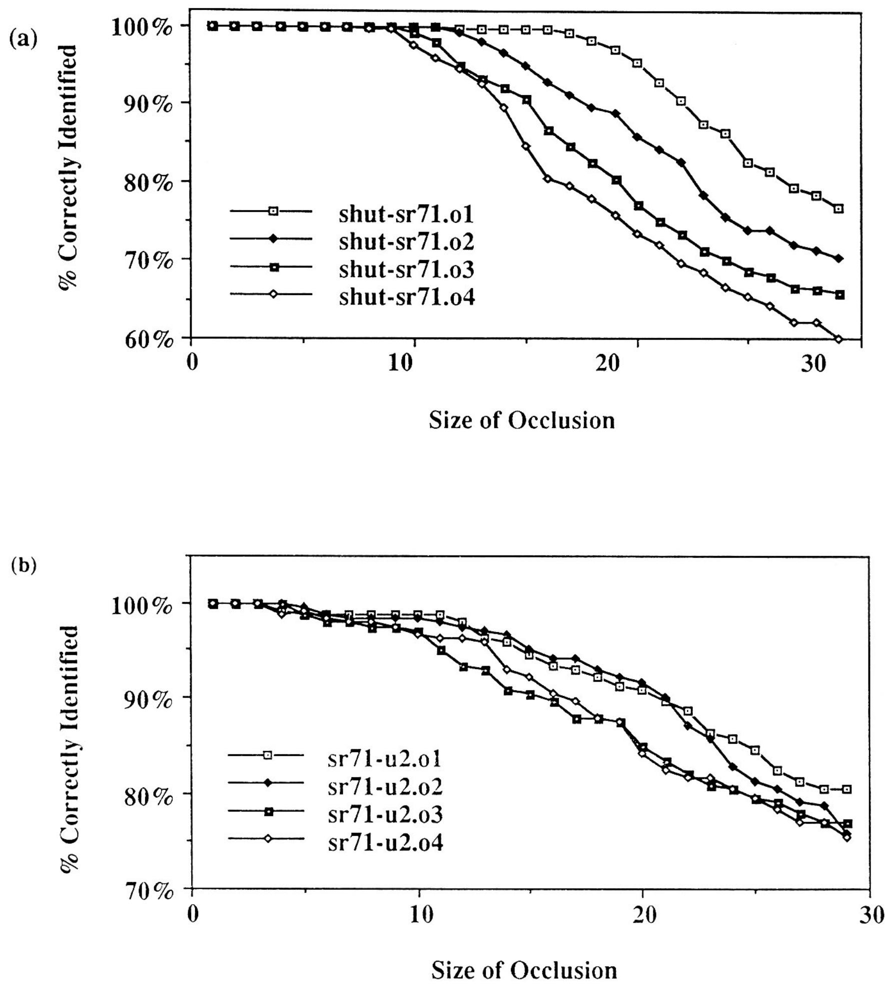


Figure 23. Tolerance of HONNS with respect to occlusion. (a) Shuttle/SR-71 discrimination problem; (b) SR-71/U-2 discrimination problem. Each graph shows the recognition accuracy for images generated as in figure 22.

approximately 90% of the test images correctly, including those shown in figure 26. In the instances where the objects were not recognized correctly, we believe the distortions introduced by the uneven lighting conditions were the primary cause.

6. 3D object recognition

For 3D object recognition, we evaluated the performance of HONNs on three vehicle discrimination problems: Space Shuttle/X-29 aircraft, Space Shuttle/VW Bug, and X-29/VW Bug.[8] As for the 2D simulations, the training sets were generated from 8-bit gray-level 3D-

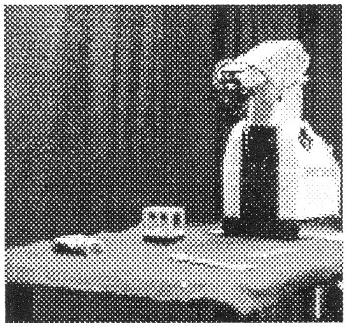


Figure 24. Photograph of the table top 5 degree-of-freedom Microbot arm and the work surface. The camera is mounted near the wrist of the arm.

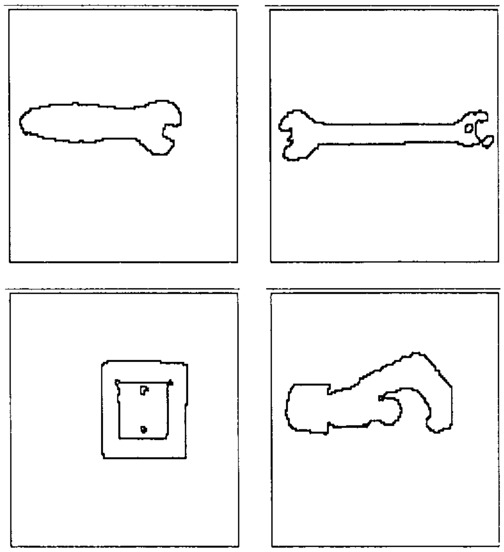


Figure 25. Robot vision experiment training images. Edge-only view of a spanner, a wrench, an assembly cube, and a hose cutter.

geometry models of the vehicles shown in figure 27, then thresholded and edge detected to produce binary contours. Since invariance to translation and in-plane rotation is built into the architecture of the network, we will discuss only the results for rotations about the roll axis and pitch axis, as depicted in figure 28.

To test the performance of HONNs to rotations about the roll and pitch axes, we simulated six instantiations of the network (X29/shuttle-roll, X29/shuttle-pitch, shuttle/VW-roll, shuttle/VW-pitch, X29/VW-roll, and X29/VW-pitch). Each instantiation was trained on 18 images generated by rotating each object about the selected axis in increments of 20° from 0° to 160° and tested on 1° rotated views from 0° to 179°. (Note that since only the

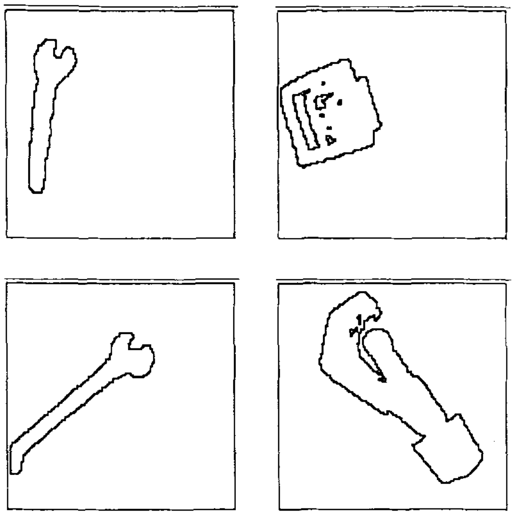


Figure 26. Examples of the views that were identified correctly by the robot vision system.

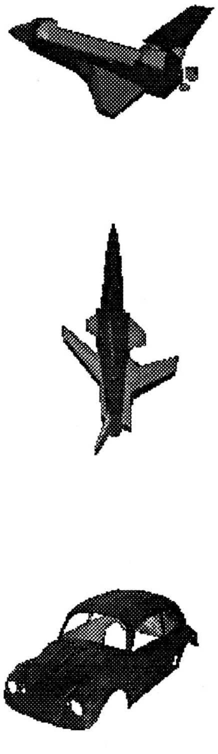


Figure 27. Eight-bit gray-level 3D-geometry models from which the training images are generated: (a) Space Shuttle orbiter; (b) X-29 aircraft; and (c) VW Bug.

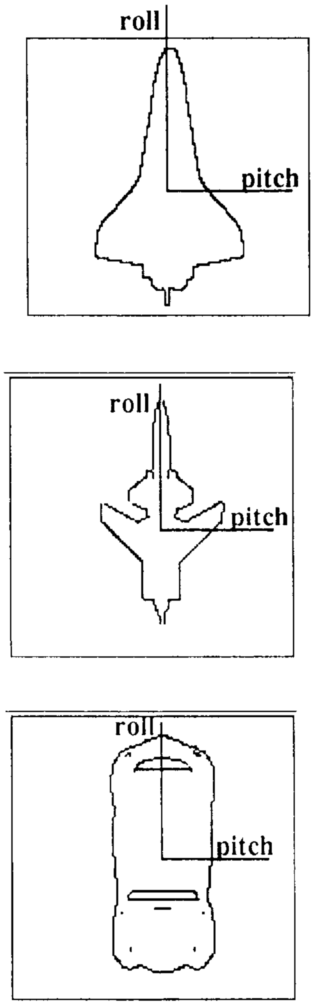


Figure 28. Edge-only view of the vehicles in figure 27, located at the origin, with the roll and pitch axes depicted.

silhouette of the objects is considered, 180° rotations are redundant.) Table 1 shows the number of passes required and the recognition accuracy results on the set of test images. For roll rotations, all three instantiations achieved 100% recognition accuracy on the test views. For pitch rotations, the instantiations of the network trained on the shuttle/VW and X29/VW pairs achieved 100% recognition accuracy on all test views, whereas the one trained on the X29/shuttle pair achieved 97.5% accuracy. The only test images classified incorrectly were the Shuttle with 89° $\leq$ pitch $\leq$ 97°. Considering that the training set included an 80° rotated image and a 100° rotated image, but not a 90° rotated image, as well as the fact that the two aircraft look very similar at a 90° pitch rotation as shown in figure 29, these results seem reasonable.

Table 1. Three-dimensional recognition accuracy.

	Shuttle/X29	X29/VW	Shuttle/VW
Roll rotation			
No. passes	40	35	213
Accuracy	100%	100%	100%
Pitch rotation			
No. passes	2473	152	209
Accuracy	97.5%	100%	100%

Note: To test the performance of HONNs to rotations about the roll and pitch axes, we simulated six instantiations of the network. Each instantiation was trained on 18 images generated by rotating each object about the selected axis in increments of 20° from 0° to 160°. "No. passes" shown represents the number of passes required to achieve 100% accuracy on the training images. The networks were then tested on 1° rotated views from 0° to 179°. "Accuracy" presents each network's performance.

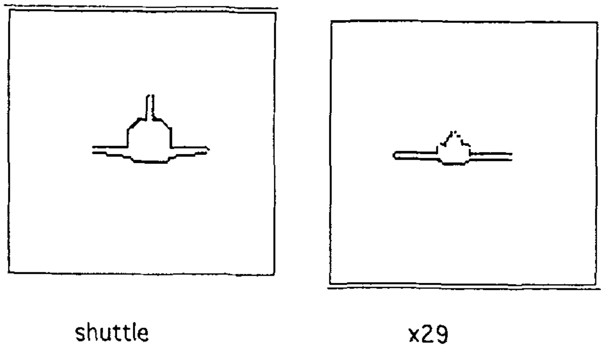


Figure 29. The 90° pitch-rotated view of (a) a Space Shuttle orbiter and (b) an X-29 aircraft.

7. Comparison with other methods

For 2D object recognition, we compared HONNs to both first-order multi-layered backprop-trained neural networks and a symbolic learning algorithm, ID3 (Quinlan, 1986). The three approaches were compared in terms of training time, training set size, and recognition accuracy.

Our simulation results show that HONNs are superior to multi-layered first-order backprop-trained networks in terms of training time, training set size, and accuracy. Learning by backprop to distinguish between a T and a C with 80%–90% accuracy, invariant to just translation and 90° rotations in a 9 × 9 pixel input field, required between 5000 and 10,000 presentations of an exhaustive training set (Rumelhart et al., 1986). A third-order network learned to distinguish between a T and a C with 100% accuracy, invariant to translation, in-plane rotation by any angle, and scale to a factor of three in just 10 passes.

HONNs also outperform ID3 in terms of recognition ability (Spirkovska & Reid, 1990). For instance, on the T/C recognition problem, ID3 was unable to learn to distinguish between the two objects simultaneously invariant to scale, translation, and in-plane rotation. Additionally, HONNs need to be trained on just one view of each object, whereas ID3 requires a large, if not exhaustive, training set. Thus, if training data are difficult to obtain, HONNs have an advantage over ID3. The comparison results for training time were inconclusive because ID3 was unable to learn to distinguish between any two objects simultaneously invariant to the three transformations.

Since the 2D object recognition task is a component of the 3D recognition task, these advantages carry over to the 3D object recognition domain. In addition, HONNs compare very favorably against a number of other approaches for 3D object recognition. We will discuss two approaches: optical correlation using binary synthetic discriminant filters and nearest-neighbor classification based on a set of normalized Fourier descriptor features.

One demonstrated approach for 3D object recognition is optical correlation using binary synthetic discriminant function (BSDF) filters (Jared & Ennis, 1989). Optical correlation is based on the Fourier transform operator, which is invariant over the object's position in the input field. Thus translation invariance is built into the architecture of the system. Further, BSDF filters can be designed to be invariant to either the size, in-plane orientation, or out-of-plane orientation of a target object. The design procedure for BSDF filters begins with a set of centered training images spanning the desired invariant feature range. A single filter can be designed to be invariant over a maximal range determined by the type of transformation it covers and the similarity of the in-class and out-of-class objects it is required to distinguish between. Experimentally, it has been shown that to distinguish between a Space Shuttle Orbiter and an F-18 aircraft, a single filter can be designed to cover an invariance range of 75° for in-plane rotations using training images in 5° increments or 20° for out-of-plane rotations using training images in 2°-3° increments (Reid et al., 1990a). For more similar objects, such as an open-end wrench vs. a box-end wrench, the interval between training images must be decreased to as little as 1° of rotation. Since a single filter cannot be designed to cover the entire 360° range for either in-plane or out-of-plane rotations, a series of filters is arranged in a tree format (Reid et al., 1990b), each filter designed to cover a limited subrange, as shown in figure 30.

Comparing this method with HONNs, it is evident that HONNs require a much smaller set of training images. Both systems are very accurate on noise-free images. However, the optical correlator may be superior in the presence of white noise or occlusions. Also, the optical correlator can recognize target objects in a cluttered scene, whereas HONNs require a segmented scene with only one object in the input field.

Another model that has recently been demonstrated for 3D aircraft recognition is based on a fast near-neighbor search (FNNS) of Fourier descriptors (Chen & Ho, 1991). This model uses the traditional approach of feature extraction followed by classification. The features used by the model are a set of Fourier descriptors proposed by Kuhl and Giardina (1982) and modified to solve problems that arise in which normalization is not uniquely determined and where small perturbations may yield different results. The classification step is implemented using a faster implementation of the well-known nearest-neighbor classification rule. The increase in speed is achieved by reducing the size of the library that must be searched for each unknown projection. To attain high recognition accuracy,

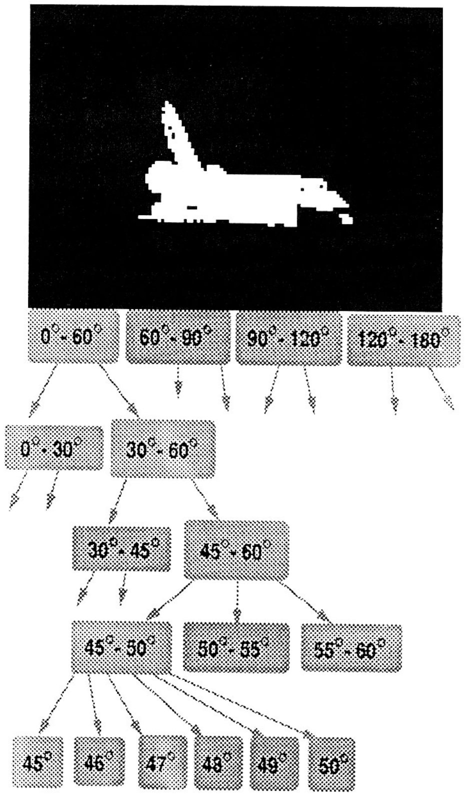


Figure 30. Database of optical filters designed to determine the out-of-plane orientation of a Space Shuttle Orbiter to within a 5° range.

this method must be applied to high library projection densities. The method achieved 98% accuracy on a test set of 50 views of five aircraft models given a density of 5° for both roll angle and pitch angle (Chen & Ho, 1991). The recognition time of the method is not specified precisely, but is categorized as slower than a decision tree method.

HONNs require a much smaller set of training images than the FNNS method to achieve comparable or better recognition accuracy. Based on Chen and Ho's (1991) claim that the FNNS method is slower than a decision tree method and our experiments showing that HONNs are as fast or faster than ID3, we conclude that HONNs are also faster than the FNNS method.

8. Conclusion

Thus far in our research, we have shown that coarse-coded HONNs are robust for 2D object recognition in both noise-free settings and in the presence of white noise and occlusion, can accomplish a subset of the 3D object recognition task based on a small set of out-of-plane rotated views, and are better than a variety of other methods in terms of training set size, recognition accuracy, and recognition time. Specifically, because the invariances are built into the architecture, high recognition accuracy of transformed views is achieved very quickly, although the network is trained on just one view of each object it is required to distinguish between. Also, once the network is trianed, it recognizes a transformed view of the training objects very quickly, on the order of a second on a Sun SPARCstation.

In continuing research, we will examine the performance of HONNs for 3D object recognition in non-ideal situations, such as with white Guassian noise and occlusion, and determine the ability of HONNs to learn to distinguish between a larger set of target objects invariant to scale, translation, in-plane rotation, and simultaneous roll and pitch rotation.

Notes

1. A fourth-order or higher network can also be used, but as will be discussed under the limitations of higher-order nets, it is advantageous to use the smallest order that provides the necessary invariances.
2. For images that are either less alike or more alike than T and C, a different bound may be necessary.
3. We used the regions 4–8, 29–37, and 55–60 for a total of 5,529,600 interconnections.
4. An input field resolution of 4096 × 4096 was also achieved by using 273 fields of 16 × 16 coarse pixels.
5. Note that this is not a limitation of the coarse-coding scheme itself. If a 4096 × 4096 pixel image could be stored on disk, using the intersection of fields approach to coarse coding, we could represent it as 228 fields of 18 × 18 coarse pixels, requiring only 5.6 MB memory.
6. The SR-71 vs. Space Shuttle problem required two passes through the training set, while the SR-71 vs. U-2 required six passes.
7. For values falling outside [0, 255], the modified gray scale value was rounded to the nearest in-range value.
8. The objects were chosen from our very limited database of 3D-geometry models.

References

Casasent, D., & Chang, W.-T. (1985.) Parameter estimation and in-plane distortion invariant chord processing. SPIE, *579*, 2-10.

Chen, Z., & Ho, S.-Y. (1986.) Computer vision for robust 3D aircraft recognition with fast library search. *Pattern Recognition*, *24*, 375–390.

Giles, G.L., & Maxwell, T. (1987.) Learning, invariances, and generalization in high-order neural networks. *Applied Optics*, *26*, 4972–4978.

Giles, G.L., Griffin, R.D., & Maxwell, T. (1988.) Encoding geometric invariances in higher-order neural networks. *Neural Information Processing Systems, American Institute of Physics Conference Proceedings*, (pp. 301-309).

Haberman, R. (1983.) *Elementary applied partial differential equations*. Englewood Cliffs, NJ: Prentice-Hall.

Hsu, Y.-N., Arsenault, H.H., & April, G. (1982.) Rotation-invariant digital pattern recognition using circular harmonic expansion. *Applied Optics*, *21/22*, 4012–4015.

Hu, M. (1962.) Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, *IT-8*, 179–187.

Jared, D.A., & Ennis, D.E. (1989.) Inclusion of filter modulation in synthetic-discriminant-function construction. *Applied Optics*, *28*, 232–239.

Kuhl, F.P., & Giardina, C.R. (1982.) Elliptic Fourier feature of a closed contour. *Computer Vision, Graphics, and Image Processing*, *18*, 236–258.

Pitts, W., & McCulloch, W.S. (1947.) How we know universals: The perception of auditory and visual forms. *Bulletin of Mathematical Biophysics*, Chicago: University of Chicago Press, *9*, 127–147.

Quinlan, J.R. (1986.) Induction of decision trees. *Machine Learning*, *1*, 81–106.

Reid, M.B., Spirkovska, L., & Ochoa, E. (1989.) Rapid training of higher-order neural networks for invariant pattern recognition. *Proceedings of Joint International Conference on Neural Networks* (Vol. 1, pp. 689–692), Washington, D.C.

Reid, M.B., Ma, P.W., Downie, J.D., & Ochoa, E. (1990a.) Experimental verification of modified synthetic discriminant function filters for rotation invariance. *Applied Optics*, *29*, 1209-1214.

Reid, M.B., Ma, P.W., & Downie, J.D. (1990b). Determining object orientation with a hierarchical database of binary synthetic discriminant function filters. *Japanese Journal of Applied Physics*, *29*, 1284–1286.

Rosenfeld, R., & Touretzky, D.S. (1988.) A survey of coarse-coded symbol memories. *Proceedings of the 1988 Connectionist Models Summer School*, Carnegie Mellon University (pp. 256–264).

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986.) Learning internal representations by error propagation. In *Parallel Distributed Processing* (Vol. 1). Cambridge, MA: MIT Press.

Spirkovska, L., & Reid, M.B. (1992.) Application of higher-order neural networks in the PSRI object recognition domain. In B. Soucek and the IRIS Group (Eds.), *Fuzzy, holographic, invariant and parallel intelligence: The sixth generation breakthrough*. New York: Wiley.

Spirkovska, L., & Reid, M.B. (1990.) An empirical comparison of ID3 and HONNs for distortion invariant object recognition. *Proceedings of the Second International Conference on Tools for Artificial Intelligence* (pp. 577–582). Washington, D.C.

Sullins, J. (1985.) Value cell encoding strategies (Technical report TR-165). Computer Science Department, University of Rochester, Rochester, NY.

Troxel, S.E., Rogers, S.K., & Kabrisky, M. (1988.) The use of neural networks in PSRI recognition. *Proceedings of Joint International Conference on Neural Networks* (pp. 593-600). San Diego, CA.

Received September 24, 1990
Accepted April 14, 1992
Final Manuscript December 15, 1992