Equilateral Triangles: A Challenge for Connectionist Vision

Subutai Ahmad[1] and Stephen Omohundro

International Computer Science Institute, Berkeley, CA.

ahmad@icsi.berkeley.edu, om@icsi.berkeley.edu

ABSTRACT

In this paper we explore the problem of dynamically computing visual relations in a connectionist system. The task of detecting equilateral triangles from clusters of points is used to test our architecture. We argue that this is a difficult task for traditional feed-forward architectures although it is a simple task for people. Our solution implements a biologically inspired network which uses an efficient focus of attention mechanism and cluster detectors to sequentially extract the locations of the vertices.

INTRODUCTION

Consider the visual task of determining whether a set of three point clusters form an equilateral triangle. People are very good at solving this kind of problem, but a connectionist implementation is not obvious. The difficulties posed by this problem are common to a wide variety of visual tasks and so we have used it as a touchstone against which to test visual neural architectures. In this paper we describe some of the fundamental operations it seems to require, biologically plausible neural implementations of those operations, and a computer simulation which combines them into a complete system.

The most straightforward visual neural representations assign a distinct unit to each visual pattern that must be classified. Unfortunately, the space of possible triangles is much too large for this kind of approach to be biologically possible. The optic nerve consists of about a million fibers from each eye, and so we consider square images which are a thousand pixels on a side. Since each of the three vertices can occupy any of these pixels, the total number of possible triangles in such an image is about $1000^6=10^{18}$. A brute force representation would require about a million times as many neurons as we have in our entire brain for just this one task. Just restricting the units to represent the set of equilateral triangles would still require about 10^{12} units.

If coarse coded representations are used, these numbers can be reduced somewhat, but many more examples will still be needed to learn to properly classify equilateral triangles than is biologically possible. The spatial relationships which define equilateralness will have to be discovered for each position, scale, and orientation of the triangle. Techniques have been proposed for introducing translation and rotational invariance into networks (Giles *et. al.*, 1987) which eliminate the need to include feature detectors at every location. Unfortunately these methods require that every unit have a large (quadratic) number of connections with complicated weight linkages between them. Furthermore, positional information is lost in these representations - one cannot retrieve the location and orientation of the objects in the image.

These difficulties would disappear if we could represent a triangle directly by the real valued coordinates of its vertices, say in the activations of 6 units. In this representation it is easy to construct units which compute the distance between a pair of points. With this setup, the network would only need to learn a classification function based on distances between points. This is a significantly easier task - the kind of task that simple backpropagation networks have been successful at. The main difficulty, of course, is to transform the repre-

1. The first author is also a graduate student in the Department of Computer Science at the University of Illinois at Urbana-Champaign.

sentation from a set of pixel values to a set of vertex coordinates.

Ullman (1987) has argued that many high-level visual tasks may be implemented as "visual routines". The idea is that a high-level system solves a visual task by choosing and combining a set of primitive visual operations. We will see that this framework is well-suited to our task, but that a plausible connectionist implementation is non-trivial. Later sections describe our solution to the equilateral triangle task with primitives for detecting and remembering the locations of clusters of points based on a focus of attention mechanism.

EVIDENCE FOR SEQUENTIAL VISUAL PROCESSING IN THE BRAIN

There is a large body of psychology literature which supports Ullman's idea of sequential visual processes. One of the best examples is the work by Treisman and her colleagues (Treisman & Gormican, 1988). Their work suggests that certain simple visual tasks are performed by people in parallel (response time is independent of the number of objects in the image) whereas other tasks require serial processing (response time is linear in the number of objects). Jolicoeur *et al.* (1986) have provided further evidence that people use sequential processes for certain visual tasks. They find that the time to report whether two stimuli lie on the same curve increases linearly with the distance between them along the curve. The presentation time is too short for saccadic eye movements.

There have also been recent neurophysiological results suggesting that neurons can dynamically change their response properties. Moran & Desimone (1985) report evidence that the sizes and locations of the receptive fields of certain neurons in the monkey visual cortex (Area V4) change with the task that the animal is trying to accomplish. This kind of behavior is suggestive of selective attention mechanisms which allow the processing of a high level system to be devoted to different regions of sensory input at different times.

NEURALLY PLAUSIBLE MECHANISMS FOR SEQUENTIAL VISUAL PROCESSING

Both of these lines of evidence suggest that serial processes play an important role in visual processing. (Ullman, 1984) suggests some specific processes which might be useful. Among them are primitives for deciding which portions of the image are relevant, selecting out these sections, and storing their locations for later processing. Given these primitives, a possible solution to our triangle problem is a system which sequentially selects and stores the location of each of the three clusters. Once the coordinates of the triangle are directly available, the distance between vertices can be explicitly computed. With this information the learning becomes trivial the network just has to learn the equality function with three inputs. In the next few sections we will describe a connectionist implementation of a fast focus of attention mechanism as well as mechanisms for detecting and storing cluster locations.

Locally Tuned Receptive Fields

In this section we describe a mechanism by which linear threshold units can give a localized response in a feature space. The following fact is exploited: if one maps the points in $\Re^{n-1}$ onto the paraboloid defined by $z = \sum_{i=1}^{n-1} x_i^2$, then the intersection of a hyperplane in $\Re^n$ with this paraboloid projects onto a sphere in $\Re^{n-1}$.

Thus there is a mapping between planes in $\Re^n$ and spheres in $\Re^{n-1}$. To select a set of points which lie within a sphere in some space one just has to project the points onto the paraboloid and slice it with the plane corresponding to the sphere. Points which lie "beneath" the plane are within the sphere. Figure 1(a) illustrates this for $\Re^2$. Notice that the computation of a threshold unit is exactly that of deciding on which side of a hyperplane an input point lies. To encode circular receptive fields with threshold units, you just need to include an extra input: the sum of the squares of all the other inputs. An equation of the form:

$$-(\sum w_i x_i + \sum x_i^2 + const) > 0 \qquad (1)$$

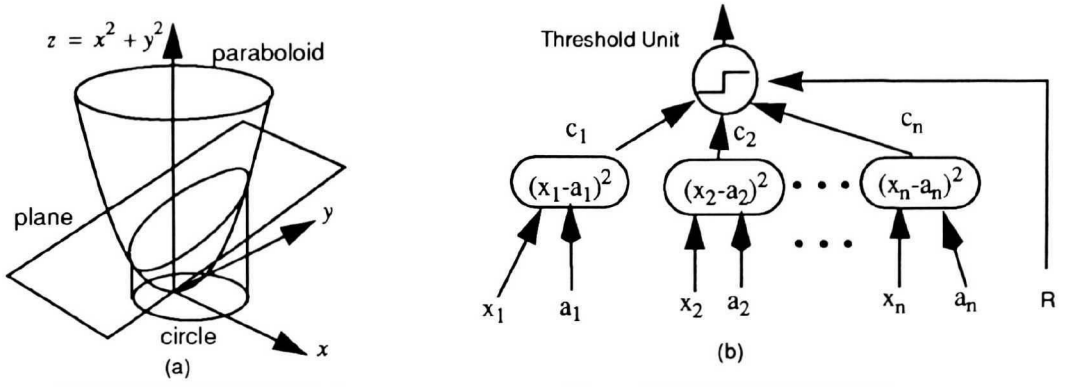


Figure 1(a) The plane intersects the paraboloid in a curve which projects to a circle. (b) Architecture of threshold unit computing the intersection shown in (a).The x_i's and a_i's are inputs to the system. The c_i's are weights denoting the stretch along each axis.

will be positive if $\dot{x}$ lies within a spherical volume determined by the weights and constant.

The above method creates circular receptive fields with hard boundaries. A smooth boundary with a flat top may be obtained by using a sigmoid instead of a threshold function. The steepness of the sigmoid (its gain) will then control the steepness of the receptive field boundary. Non-circular receptive fields are also possible by changing the nature of the non-linearity. Elliptical receptive fields may be obtained by using the paraboloid $z = \sum_{i=1}^{n-1} c_i x_i^2$ where c_i denotes the amount of stretching along each axis. In principle arbitrary shapes can be obtained by appropriately choosing the non-linearity.

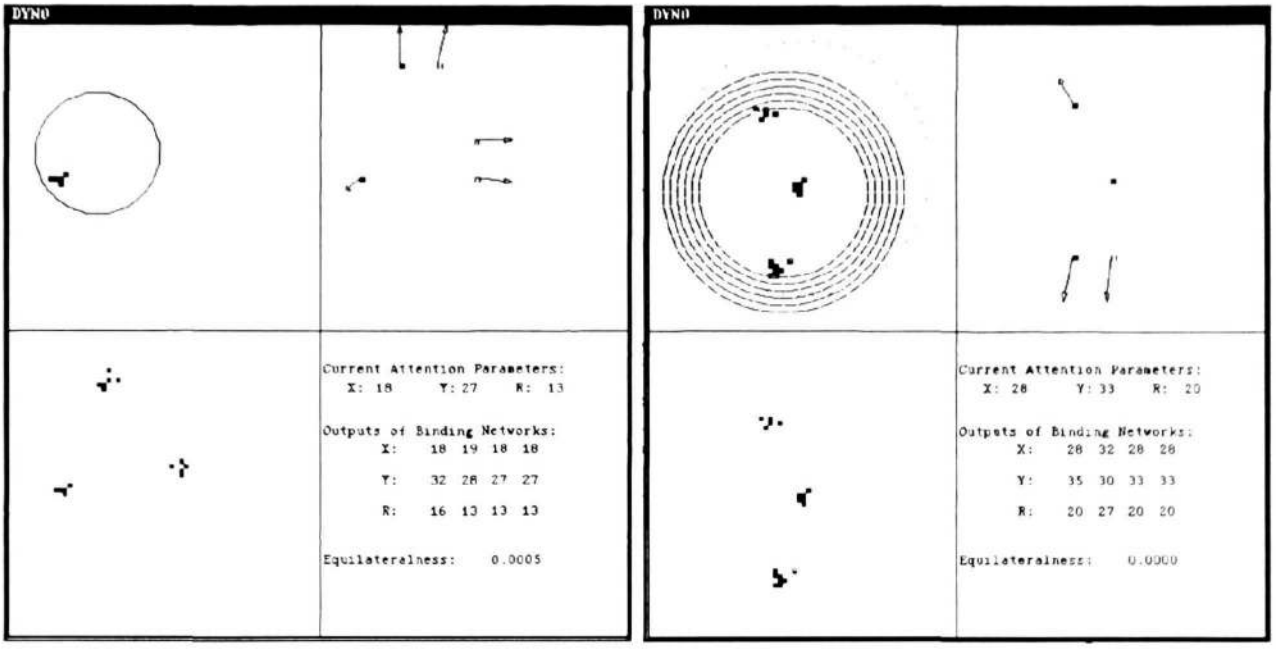
Dynamic Receptive Fields

In addition to being able to select a portion of the input space, we need the ability to shift the location and size of the receptive field around quickly in response to changing demands. In our model there are basically two ways of doing this. The first method involves changing the slope of the hyperplane. In Figure 1(a) note that as the slope increases the center of the projected circle will shift to the right. For any sphere it is possible to compute the coefficients of the hyperplane which produces that sphere. In a threshold unit, changing the slope of the hyperplane corresponds to changing the weights of the inputs. One of these units could eventually learn the correct position of its receptive field. However the time scale for weight changes is too slow for dynamic computation.

Another way to alter the sphere is to shift the paraboloid itself, by computing $z = \sum_{i=1}^{n-1} c_i (x_i - a_i)^2 + r$. This moves the paraboloid a distance a_i along dimension i (changing the location of sphere) and a distance r along the z-axis (changing the radius of the sphere). If the a_i's and r are available as input then the receptive field can be changed an arbitrary amount in one time step. Figure 1(b) shows how such a unit would be configured. For each input dimension there is a sub-unit which computes the square. ((Suarez & Koch, 1989) present a neurally plausible mechanism for computing a quadratic.) The outputs of these units are fed into a threshold (or sigmoid) unit. The net effect is that the threshold unit will respond only when the input vector $\dot{x}$ lies within the spherical receptive field determined by $\dot{a}$ and r.

Focus Of Attention With Value Coded Units

So far we have assumed an n-dimensional input space that is encoded as n analog signals. In our triangle task however, we have to implement a circular patch in a 2-dimensional retina. The units in this representa-



(a) (b)

Figure 2 Examples of the system behavior for two different 64x64 images. In both displays the lower left quadrant shows the image; the upper left quadrant shows the output of the gate units. The error vectors are displayed on the upper right and the outputs of various units are shown in the lower right. (a) shows a snapshot of the system with the focus of attention near one of the vertices. (b) shows the dynamic scaling behavior as the focus tries tofit the cluster of points within it.

tion are laid out on a flat sheet, with each unit explicitly encoding a point in the space.

To create dynamic receptive fields here, we construct a gating layer, with one "gate unit" per input unit. Each gate unit receives three global inputs: A_x, A_y, and A_r representing the parameters of the focus of attention. The gate units are basically the same as the localized units described above with one small modification: x_1 and x_2 are fixed for each gate unit. To encode this we have two separate connections from the input unit to the gate unit. The weights of these connections are fixed as x_1 and x_2. If the input unit fires, the threshold unit will fire only when its input unit is within the circle determined by A_x, A_y, and A_r. The effect is a layer of units which filters the input image according to a global control signal. The system can select any circular portion of the image in one time step. This is quite different from Mozer's implementation (Mozer, 1988) where the network has to iteratively settle on a solution, or Chapman's implementation (Chapman, 1990) which takes logarithmic time per selection. The hardware required to implement this is minimal: 8 extra connections per input unit. It is also fairly easy to extend our mechanism to allow foci of different shapes once the parameters are available.

Figure 2 shows the graphical output of our simulator. In each display, the lower left quadrant displays the current input image (3 point clusters). The upper left quadrant shows the output of the gate units. The circle shows the focus of attention represented by the 3 parameters. In Figure 2 (a) note that only the activity within the focus is allowed to propagate.

Deciding Where To Focus ... And How To Get There.

We need a mechanism to cause the focus of attention to sequentially fixate on the interesting portions of the image. There are two different cases to consider: 1) there is already a cluster within the focus of attention, and 2) the clusters are outside the focus. In the first case our system fixates on the center of mass of the cluster of points and wraps the focus of attention around the cluster. Once this is accomplished, the parameters of the focus of attention provide an accurate estimate of the location and size of the cluster. To deal with the

second case we have a system which receives input directly from the image and provides a rough estimate of the location of the next cluster to visit. These two mechanisms are described in the following sections.

Clusters Within The Focus Of Attention

The center of mass of a cluster, (C_x, C_y), is the average of the x and y coordinates of the active points:

$$C_x = \frac{\sum_i X(i)\, a_i}{\sum_i a_i} \text{ and } C_y = \frac{\sum_i Y(i)\, a_i}{\sum_i a_i} \tag{2}$$

where $X(i)$ and $Y(i)$ denote the x and y coordinates of the i'th unit and a_i denotes its activity. $\sum a_i$ can be computed by a unit which receives input from all gate units with a weight of 1. To compute $\sum X(i)\, a_i$ and $\sum Y(i)\, a_i$ we include two units with links to every gate unit. The weights from the i'th unit to each of these two units are $X(i)$ and $Y(i)$, respectively. A weighted sum of the incoming activity computes the appropriate value. The sums are divided by $\sum a_i$ to calculate C_x and C_y as the values of two units. These values are fed into two other units which compute the difference between the cluster center and the attention parameters: $(C_x - A_x, C_y - A_y)$. Units representing A_x and A_y receive as input this difference as well as their own output. By computing the sum of all their inputs, these units keep the field of attention continually centered on the cluster of points within the focus.

To get an estimate of the size of the cluster we include a scaling unit which continually adjusts the size of the focus of attention to match the size of the set of points within it. The rule is that as long as $\sum a_i$ remains constant, the scaling unit decreases A_r by a small amount. If the sum decreases, indicating that the scale has become too small, the unit increases A_r slightly and stops.

Figure 2 (b) shows the focus of attention changing to adjust to the cluster within it. The dotted circle represents the initial location. The set of concentric bands show successive steps as the focus of attention decreases and shifts its location to fit the cluster inside.

Clusters Outside The Focus Of Attention

The mechanism described above continually fine tunes A_x, A_y, and A_r to match the cluster of points within the focus but does not give us a way to fixate on clusters outside the focus. To do this we include a coarse grid of units which receives input directly from a circular patch in the image. At each grid location there are three outputs to consider. The first two outputs encode the difference between center of mass of the cluster of points within their receptive fields and the point (A_x, A_y). The third output is simply the number of active points within its receptive field, passed through a sigmoid. Thus each grid location encodes an "error" vector for adjusting the focus of attention, and a confidence value from 0 to 1 indicating the importance of the cluster. These error vectors are continually updated to compensate for changes in A_x and A_y. This error vector representation was inspired by the mechanism in the monkey superior colliculus for controlling eye saccades as reported by (Sparks, 1986).

The upper right quadrant of Figure 2 (a) and (b) shows the outputs of the error units. Each arrow represents the error vector at that location. The shaded square represents the confidence value - the darker the square the higher the confidence. (Only those locations whose confidence value is higher than 0.2 is displayed.)

To sequentially process each cluster in the image the system has to repeatedly select the largest confidence value, inhibit the corresponding unit, and send the error vector to the system controlling A_x and A_y. We are currently investigating several mechanisms for choosing the largest value. One could construct a winner-take-all network with competing confidence units such that the system settles into a state where only one

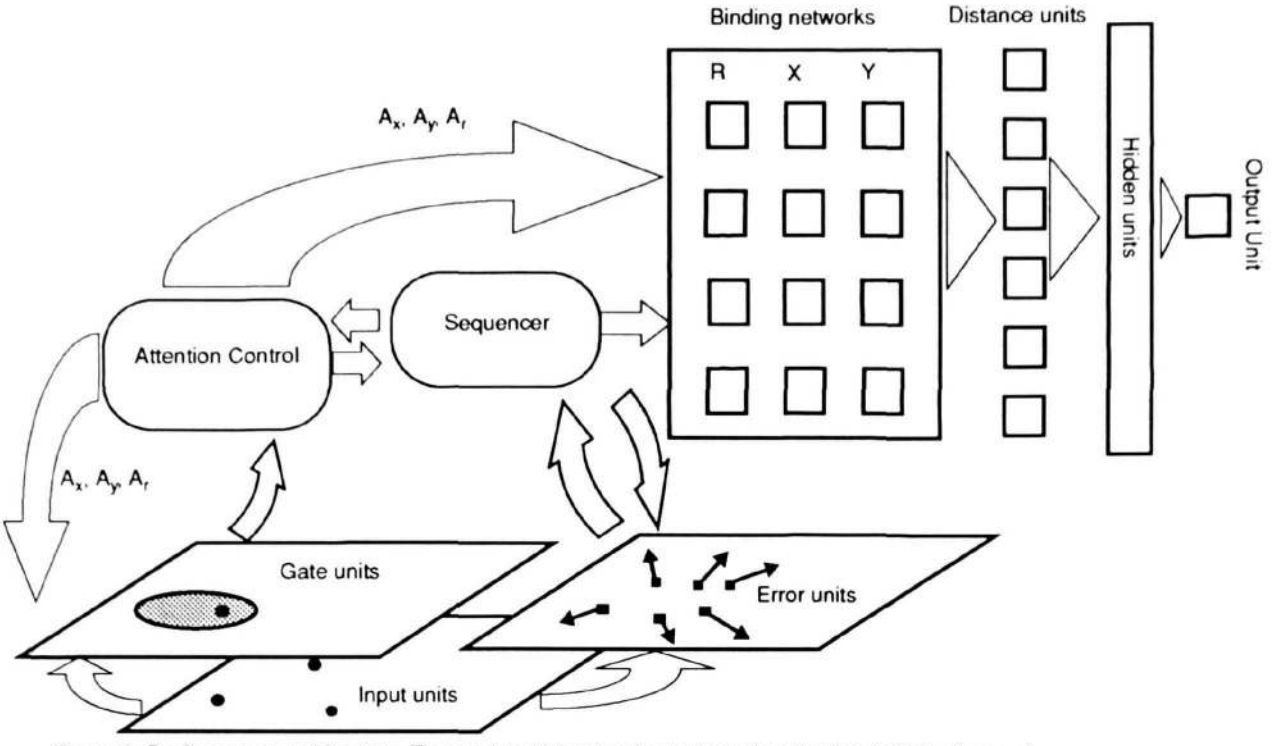


Figure 3. Basic system architecture. The module "Attention Control" continually tries to fit the focus of attention to the points within it. The "Sequencer" updates the focus of attention to visit all the clusters in sequence and also sends the control signals to the binding network to store the successive locations.

unit is active. However these networks can take a relatively long time to settle. It is also quite difficult to find the correct set of inhibitory weights to create a robust winner-take-all network. Another alternative is to construct a log-depth network of threshold units to explicitly compute the maximum, however this is equally unappealing. We currently assume a max finding unit but are studying a temporal representation which makes this computation biologically plausible.

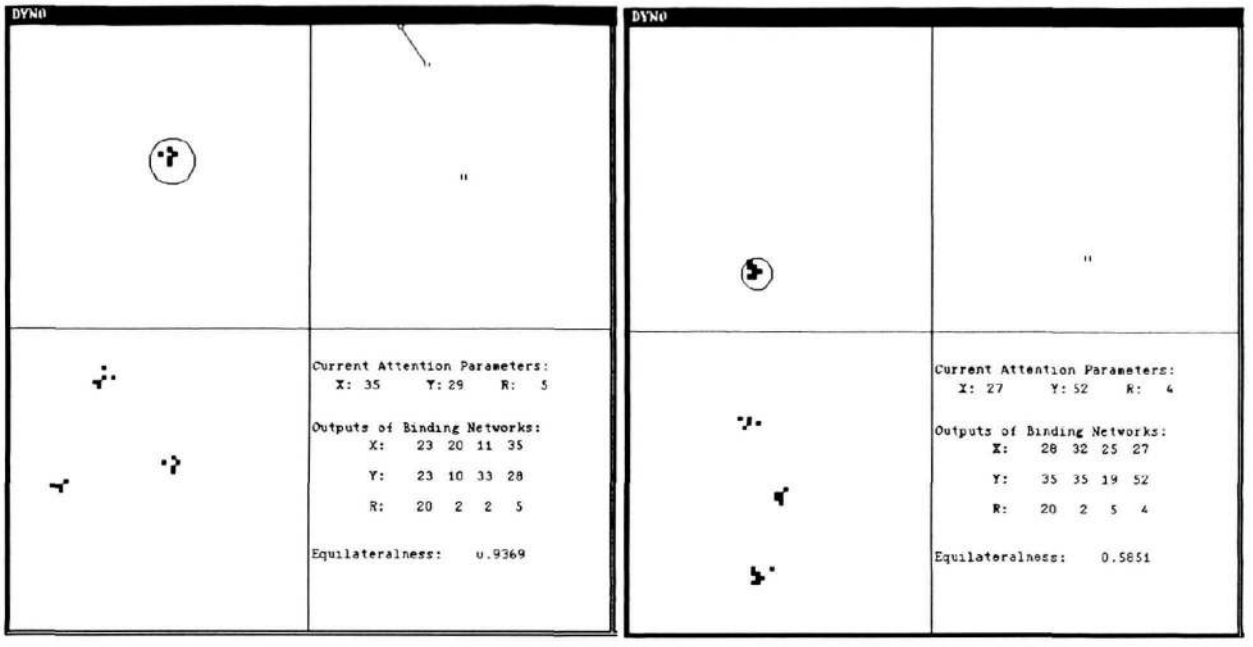
Storing Locations

As the network visits each vertex it should store the values of A_x, A_y, and A_r whenever the focus of attention stabilizes. We accomplish this with small recurrent networks (3 units) for each value that needs to be stored. Each of these networks continually tracks a particular unit (one of A_x, A_y, and A_r) until a control signal is sent, whereupon it freezes the output to be the current value of the unit. This is done by including a hidden unit which receives a strong inhibitory link from its control unit. When the control is off, it computes the difference between the value of the assigned unit and the current output and sends it to the output unit. An excitatory link from the control unit to itself ensures that once the control unit has fired, it stays on, preventing further adjustments. Three of these "binding networks" are used for each set of parameters that are stored.

SIMULATION

Figure 3 shows a schematic of the whole architecture. The module which controls the attention field is an autonomous network that continually attempts to wrap the focus around the points within its field of view. The sequencer waits until the focus of attention has stabilized and then does two things: it transmits a control signal to the binding networks to store the current parameters and updates the focus of attention to the location of the next cluster.

Assuming that the system starts with a focus of attention covering the entire image plane, the network first wraps the focus around the triangle and then sequentially visits and stores the locations of the three vertices.



(a) (b)

Figure 4. The result after parsing the two images shown in Figure 2.

Once this is done, the first set of bindings encode the position and scale of the triangle. The other nine bindings encode the positions and scales of the three vertices in the order that they were processed. A set of distance units then explicitly computes the six possible distances between the four stored locations. A standard feedforward network with one layer of hidden units is used to compute the final output. Space limitations prevent a more detailed explanation of the control structure. It is worth noting however that every aspect of it is implemented with units computing simple functions (except for the max function as described above). Details on the exact set up can be found in (Ahmad & Omohundro, 1990).

For the simulations in this paper we used images with 64x64 pixels. We generated a training set consisting of random triangles (approximately 50% of which were equilateral) with Gaussian noise added around each vertex. For each triangle the focus of attention was initialized to cover the entire image plane. The system was allowed to run until 4 control signals were generated. The outputs of the distance units were then used as inputs to train the backprop network. The teacher signal used was $1 - \frac{|l_1 - l_2| + |l_2 - l_3| + |l_1 - l_3|}{l_1 + l_2 + l_3}$, where l_i is the length of the i'th side. This is a function which is 1 for equilateral triangles and degrades gradually to 0 as the triangles deviate from equilateralness. With a training set of 100 triangles the network score was consistently greater than 0.9 for equilateral triangles. The specifics of the learning are not crucial to this paper, however note that the number of training examples needed would *not* increase if we increased the image size.

Figures 4 (a) and (b) show the state of the network after parsing the two triangles in Figure 2. The system correctly classified the left triangle as being equilateral and the right one as not being equilateral. The outputs of the binding networks show the vertex coordinates (in pixels) that were discovered by the network. The current implementation requires about 50 seconds on a Sun 4 to extract the coordinates of one triangle. By far the majority of this time is taken up by the scaling process.[1] This is because in our simulation we decrease the scale by a small constant at each time step. Starting off with a large focus of attention, a large number of steps may be necessary before the scale matches the cluster. We are investigating an implemen-

1. We don't actually need the scaling part for the triangle task however a general purpose cluster detector should have the ability to extract cluster sizes.

tation in which the scale shrinks continuously at a time scale faster than the interspike interval time of individual neurons.

DISCUSSION

It is interesting to speculate as to how mechanisms like the ones we have described would fit into a general purpose vision system. The library of primitives must be expanded to handle more features of realistic images. We will need primitives for intelligent processing of curves, regions, and shapes. Another issue is the large number of different representations that are formed in parallel in the brain. A mechanism for integrating the information available in these representations will be necessary. Since the focus of attention mechanism we described can be implemented in any topological space, in principle it should provide a good way of isolating exactly the subsets that are relevant.

Another major issue that we have not addressed is how to compile these visual primitives to accomplish a dynamically specified task. Such a system would presumably need a language for specifying the tasks. To translate the specification of the task into the appropriate primitives requires that the intermediate representations of the visual primitives and the descriptive language should be very similar. (Feldman *et. al.*, 1990) discusses some of these issues in the context of a novel approach to language acquisition.

In conclusion, the main point of this paper has been to demonstrate neurally plausible mechanisms for performing sequential visual computations. There is evidence from psychology and neurophysiology that biological organisms actually implement such routines at an early processing level. We have described an efficient implementation of various primitives within a connectionist framework, and have used them to extract image properties that are otherwise extremely inefficient to represent.

REFERENCES

Ahmad, S., & Omohundro, S. (1990). A Connectionist System for Extracting the Locations of Point Clusters. Technical Report TR-90-011, International Computer Science Institute, Berkeley, CA.

Chapman, D. (1990). *Instruction Use in Situated Activity.* Ph.D. Thesis, Massachusetts Institute of Technology.

Feldman, J.A., Lakoff, G., Stolcke, A., & Weber, S.H. (1990). Miniature Language Acquisition: A Touchstone for Cognitive Science. Submitted to the 12th Annual Conference of the Cognitive Science Society, MIT, July 1990.

Giles, C.L., Griffin, R.D., & Maxwell, T. (1987). Encoding Geometric Invariances in Higher Order Neural Networks. In "Advances in Neural Information Processing", David Touretzky, Ed. Morgan Kaufmann.

Jolicoeur, P., Ullman, S., & Mackay, M. (1986). Curve tracing: A possible basic operation in the perception of spatial relations. *Memory and Cognition*, **14** (2), 129-140.

Minsky, M. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA.

Moran, J. & Desimone, R. (1985). Selective Attention Gates Visual Processing in the Extrastriate Cortex. *Science*, **229**, March 1985.

Mozer, M. (1988). A Connectionist Model of Selective Attention in Visual Perception. University of Toronto Technical Report CRG-TR-88-4

Sparks, D. L. (1986). Translation of Sensory Signals into Commands for Control of Saccadic Eye Movements: Role of Primate Superior Colliculus, *Physiological Reviews*, **66** (1).

Suarez, H., & Koch, C. (1989). Linking Linear Threshold Units with Quadratic Models of Motion Perception. *Neural Computation*, **1** (3), pp 318-320.

Treisman, A., & Gormican, S. (1988). Feature Analysis in Early Vision: Evidence From Search Asymmetries. *Psychological Review*, **95** (1), pp 15-48.

Ullman, S. (1984) Visual Routines. *Cognition*, **18**, pp 97-159.