LETTER Communicated by David Lowe

A Neural Network Architecture for Visual Selection

Yali Amit
Department of Statistics, University of Chicago, Chicago, IL 60637, U.S.A.

This article describes a parallel neural net architecture for efficient and robust visual selection in generic gray-level images. Objects are represented through flexible star-type planar arrangements of binary local features which are in turn star-type planar arrangements of oriented edges. Candidate locations are detected over a range of scales and other deformations, using a generalized Hough transform. The flexibility of the arrangements provides the required invariance. Training involves selecting a small number of stable local features from a predefined pool, which are well localized on registered examples of the object. Training therefore requires only small data sets. The parallel architecture is constructed so that the Hough transform associated with any object can be implemented *without creating or modifying any connections*. The different object representations are learned and stored in a central module. When one of these representations is evoked, it "primes" the appropriate layers in the network so that the corresponding Hough transform is computed. Analogies between the different layers in the network and those in the visual system are discussed. Furthermore, the model can be used to explain certain experiments on visual selection reported in the literature.

1 Introduction

The issue at hand is visual selection in real, still gray-scale images, which is guided by a very specific task of detecting a "memorized" or "learned" object class. Assume no "easy" segmentation cues are available, such as color or brightness of the object, or a relatively flat and uniform background. In other words, segmentation can not precede detection. Moreover, the typical 240×320 gray-scale image is highly cluttered with multiple objects at various scales, and dealing with false positives is crucial. The aim of selection is to choose quickly a small number of candidate poses for the object in the scene, from among the millions of possibilities, taking into account all possible locations and a range of scales. Some of these candidate poses may be false and may be further analyzed for final verification, using more intensive procedures, which are not discussed here.

Various biological facts and experiments must be dealt with when constructing a computational model for visual selection.

- Selection is a very rapid process—approximately 150 milliseconds after presentation of the image (Thorpe, Fize, & Marlow, 1996; Desimone, Miller, Chelazzi, & Lueschow, 1995). Moreover of these 150 milliseconds, at least 50 milliseconds are spent on processing from the retina through the primary visual cortex V1, which is assumed to be involved with detection of edges, various local motion cues, and so forth. So very little time is left for global processing necessary to determine a candidate location of the object.
- There is a limited number of processing stages between V1 and IT, where there is clear evidence for selective responses to full objects even prior to the saccade. (See Desimone et al., 1995.) A common schematic description of these stages includes V1 simple cells, V1 complex cells, V2 cells detecting more complex structures (e.g., illusory contours, t-junctions, corners; see von der Heydt, 1995), V4 cells, and finally IT. Receptive field size increases as one moves up this hierarchy.
- The receptive fields of cells in a given column, which corresponds to approximately the same location in the visual field and to the same feature, exhibit significant deviations with respect to each other. These deviations increase with the level in the hierarchy (Zeki, 1993).
- Selection is not a very accurate process. Mistakes can be made. Often it is necessary to foveate to the location and process the data further to make a final determination as to whether the required object is indeed there.
- Selection is invariant to a range of scales and locations (Lueschow, Miller, & Desimone, 1994; Ito, Tamura, Fujita, & Tanaka, 1995).

The question is whether there are computational approaches that accommodate the constraints listed above—that is, can efficient selection be achieved at a range of scales, with a minimal number of false negatives and a small number of false positives, using a small number of functional layers? Can the computation be parallelized, making minimal assumptions regarding the information carried by an individual neuron? Such computational approaches, if identified, may serve as models for information processing beyond V1. They can provide a source of hypotheses that can be experimentally tested.

The aim of this article is to present such a model in the form of a neural network architecture that is able to select candidate locations for any object representation evoked in a memory model; the network does not need to be changed for detecting different objects. The network has a sequence of layers similar to those found in visual cortex. The operations of all units are simple integer counts. A unit is either on or off, and its status depends

on the number of on units feeding into it. Selection at a fixed resolution is invariant to a range of scales and all locations, as well as small linear and nonlinear deformations.

1.1 Overview of the Selection Algorithm. Selection involves finding candidate locations for the center of the object, which can be present at a range of poses (location, small rotations, scaling of $\pm$ 20% and other small deformations). In other words, the detection is invariant to the range of poses but does not convey information about the parameters of the pose except for location. Various options for recovering pose parameters such as scale and rotation are discussed in Amit and Geman (1999) and Amit (1998). This issue is not dealt with here.

Selection can therefore be viewed as a process whereby each location is classified as object or background. By object, we mean that an object is present at one of an admissible range of deformations at that location. There are two pitfalls arising from this point of view that should be avoided.

The first is the notion that one simply needs training samples from the object at the predetermined range of scales and samples from background images. These are then processed through some learning algorithm (e.g., tree classifiers, feedforward neural nets, support vector machines) to produce the classifier. The second is that this classifier is subsequently implemented at each location. The problem with this point of view is that it will produce a different global architecture (tree, neural net, etc.) for each new object that needs to be detected; it will require a large training set both to represent background images and to learn the required range of scales; it will not directly address the issue of efficiency in computation and resources required for detection of multiple objects. It should be noted, however, that this approach has led to successful computer vision algorithms for specific objects. For an example in the context of face detection, see Rowley, Baluja, and Takeo (1998) and Sung and Poggio (1998).

In view of these issues and in order to accommodate the constraints described in the previous section, the selection system described below is guided by the following assumptions:

- The classification carried out at each image location—object versus background—must be a very simple operation. One cannot expect a sophisticated classifier such as a heavily connected neural net to be present at each location.

- Objects are represented through spatial arrangements of local features, not through individual local features. Individual local features are insufficient to discriminate between an object and the rich structure of the background; multiple local features are also insufficient without spatial constraints on their relative locations.

- The basic inputs to the network are coarse-oriented edge detectors with a high degree of photometric invariance. Local features are defined as functions of these edges.
- Training involves determining moderate probability local features, at specific locations on examples of the object, which are registered to a reference pose in a reference grid.
- Invariance to local deformations is hard wired into the definition of the local features, not learned. Invariance to global transformations is hard wired into the computation and not learned.
- No changes need to be made to the network for detecting new objects except for learning their representation in a central memory module. The representation does not need to be distributed to all locations.
- The network codes for the continuous location parameter through a retinotopic layer, which is active (on) only at the detected locations. Individual units are not expected to code for continuous variables.

The solution proposed here is to represent objects in terms of a collection of local features constrained to lie in certain regions relative to a virtual center. In other words the spatial conjunction employed here can be expressed in a very simple *star-type* graph (see Figure 2, right panel.) Any constraints involving relative locations of pairs or larger subsets of features significantly increase the computational load or the complexity of the required parallel architecture. Selection involves finding those locations for which there is a sufficient number of such features satisfying the relative spatial constraints. This is done using a version of the generalized Hough transform (see Grimson, 1990).

The parallel architecture is constructed so that the Hough transform associated with any object can be implemented *without creating or modifying any connections*. The different object representations are stored in a central module. When one of these representations is evoked, it primes the appropriate layers in the network so that the corresponding Hough transform is computed. The network involves two levels of simple and complex type neurons. The simple layers detect features (edge and edge conjunctions) at specific locations; the complex layers perform a disjunction over a certain region. These disjunctions are appropriately displaced from the original location of the feature, enabling a simple summation to compute the Hough transform. The disjunction regions are not learned but are hard-wired into the architecture.

The features come from a predefined pool. They are constructed so as to have sufficiently low density on generic background. Training for an object class involves taking a small number of examples at reference pose (i.e., registered to the reference grid) and identifying those features from the pool that are frequent at particular locations in the reference grid. These locations implicitly define the global spatial relations. There is no spe-

cial learning phase for these relations. The object representation is simply the list of selected features with the selected locations on the reference grid.

The entire architecture uses a caricature of neurons as binary variables that are either on or off. This is clearly a very simplistic assumption, but it has several advantages. The description of the algorithm is greatly simplified, its serial implementation becomes very efficient, and it does not a priori commit to any specific assumption on the information conveyed in the continuous output of any of the neurons. On the other hand, starting from the model presented here, it is possible to incorporate multivalued outputs for the neurons at each stage and carefully study how this affects the performance, stability, and invariance properties of the system.

Two main conclusions can be drawn from the experiments and model described here.

- There is a way to create good object and background discrimination, with low false-negative and false-positive rates, in real images, with a very simple form of spatial conjunctions of local features. The local features are simple functions of the edge maps. This can be viewed as a statistical statement about objects and background in images.
- The detection algorithm has a simple parallel implementation, based on the well-known Hough transform, that exhibits interesting analogies with the visual system. In other words, efficient visual selection can be done in parallel.

The article is organized as follows. Section 2 discusses relations to other work on detection. Section 3 provides a description of the detection algorithm, the type of local features employed, and how they are identified in training. In section 4 the neural architecture for such detection is described. In section 5 the biological analogies of the model are discussed and how the architecture can be used to explain certain experiments in visual selection reported in the literature.

2 Related Work

The idea of using local feature arrangements for detection can also be found in Burl, Leung, and Perona (1995), Wiskott, Fellous, Kruger, and von der Malsburg (1997), and Cootes and Taylor (1996). In these approaches the features, or certain relevant parameters, are also identified through training. One clear difference, however, is that the approach presented here makes use of very simple binary features with hard-wired invariances and employs a very simple form of spatial arrangement for the object representation. This leads to an efficient implementation of the detection algorithm. On the other hand, the representations described in these articles are more detailed, provide more precise registration information, and are useful in the more

intensive verification and recognition processes subsequent to selection and foveation.

In Van Rullen, Gautrais, Delorme, and Thorpe (1998) a similar representation is derived. Features defined in terms of simple functionals of an edge input layer are used to represent components of the face, making use of the statistics of the edges on the training set. They model only three locations (two eyes and mouth) on the object, which are chosen by the user, training multiple features for each. In the approach suggested here, many more locations on the object are represented using one feature for each. These locations are identified automatically in training. One problem with overdedicating resources to only three locations is the issue of noise and occlusion, which may eliminate one of the three. Our representation for faces, for example, is at a much lower resolution—14 pixels between the eyes on average. At that resolution the eyes and mouth in themselves may be quite ambiguous. In Van Rullen et al. (1998) *pose-invariant* detection is achieved by having a "face neuron" for each location in the scene. This raises a serious problem of distributing the representation to all locations. One of the main goals of the architecture here is to overcome the need to distribute the representation of each new object that is learned, to a dedicated "object neuron" at each location.

The generalized Hough transform has been extensively used in object detection (see Ballard, 1981; Grimson, 1990; Rojer & Schwartz, 1992). However, in contrast to most previous uses of the Hough transform, the local features are more complex and are identified through training, not from the geometric properties of the object. It should be emphasized that this "trained" Hough transform can be implemented with any pool of binary features that exhibit the appropriate invariance properties and the appropriate statistics on object and background—for example, those suggested by Van Rullen et al. (1998).

In the use of simple and complex layers this architecture has similarities with Fukushima (1986) and Fukushima and Wake (1991). Indeed, in both articles the role of ORing in the complex cell layers as a means of obtaining invariance is emphasized. However, there are major differences between the architecture described here and the neocognitron paradigm. In our model, training is done only for local features. The global integration of local-level information is done by a fixed architecture, driven by top-down information. Therefore, features do not need to get more and more complex. There is no need for a long sequence of layers. Robust detection occurs directly in terms of the local feature level, which has only the oriented edge level below it.

3 Detection

We first describe a simple counting classifier for object versus background at the reference pose. Then, an efficient implementation of this classifier on an entire scene is presented using the Hough transform.

3.1 A Counting Classifier. Let G denote a $d \times d$ reference grid, centered at the origin, where $d \sim 30$. An image from the object class is said to be registered to the reference grid if certain well-defined landmarks on the object are located at fixed locations in G. For example, an image of a face would be registered if the two eyes and the mouth are at fixed locations.

Let $\alpha_1, \ldots, \alpha_{n_{ob}}$ be binary local features (filters) that have been chosen through training (see section 3.4) together with locations $y_1, \ldots, y_{n_{ob}}$ in G. This collection constitutes the object representation. For any $i = 1, \ldots, n_{ob}$ and any location x in a scene, we write $\alpha_i(x) = 1/0$ to indicate the presence or absence of local feature α_i in a small neighborhood of x. A threshold τ is chosen. An image in the reference grid is then classified as object if at least τ of these features are present at the correct location, and background otherwise.

Assume that conditional on the presence of a registered object in the reference grid the variables $\alpha_i(y_i)$ are independent, with $P(\alpha_i(y_i) = 1|Object) = p_o$, and conditional on the presence of a generic background image in the reference grid these variables are all also independent with $P(\alpha_i(y_i) = 1|Bgd.) = p_b << p_o$. Using basic properties of the binomial distribution, we can predict the false-positive and false-negative probabilities for given n_{ob}, τ, p_o, p_b (see Rojer & Schwartz, 1992; Amit, 1998).

3.2 Finding Candidate Poses. How can this classifier be extended to the full image, taking into account that the object may appear at all locations and under a large range of allowable transformations?

Assume all images are presented on a $D \times D$ grid L and centered at the origin, with $D \sim 300$. Thus, the reference grid is a small subset of the image grid. Henceforth y will denote locations on the reference grid, and x will denote locations in the scene or image. A pose of the object in the image is determined by a location x_c and a map $A \in \mathcal{A}$, where $\mathcal{A}$ is some subset of linear (and perhaps even nonlinear) transformations of the plane. Typically $\mathcal{A}$ accommodates a range of scales of $\pm$ 25% and a small range of rotations.

To decide whether the object is present at a location $x_c \in L$, for each transformation $A \in \mathcal{A}$, we need to check how many of the n_{ob} features α_i are present at the transformed location $x_i = Ay_i + x_c$. If at least τ are found for some A, then the object is declared present at location x_c. This operation involves an extensive search through the pose set $\mathcal{A}$ or, alternatively, verifying complex constraints on the relative locations of pairs or triples of local features (see Grimson, 1990). It is not clear how to implement this in a parallel architecture.

We simplify by decoupling the constraints between the local features. To decide if the object is present at x_c, we check for each feature α_i whether it is present at $x_i = Ay_i + x_c$ *for any* $A \in \mathcal{A}$. Namely, the map A may vary from feature to feature. In other words we count how many of the regions $x_c + B_{y_i}$ contain at least one instance of the corresponding feature α_i, where

$B_{y_i} = \{Ay_i\colon\ A \in \mathcal{A}\}$. If we find at least τ, we declare the object is present at x_c. This is summarized as follows:

(C) There exist at least τ indices $i_1, \ldots, i_\tau$, such that the region $x_c + B_{y_{i_j}}$ contains an instance of local feature α_{i_j}, for $j = 1, \ldots, \tau$.

The object model can now be viewed as a flexible star-type planar arrangement in terms of the local features, centered at the origin (see Figure 2, right panel.) This is a very simple system of spatial constraints on the relative locations of the features. Decoupling the constraints allows us to detect candidate locations with great efficiency. There is no need to proceed location by location and verify (C). Starting from the detected locations of the local features, the following Hough transform provides the candidate locations of the object:

Hough Transform

1. Identify the locations of all n_{ob} local features in the image.
2. Initialize $D \times D$ arrays Q_i, $i = 1, \ldots, n_{ob}$ at 0.
3. For every location x of feature α_i set to 1, all locations in the region $x - B_{y_i}$ in array Q_i.
4. Choose locations x_c for which $\sum_{i=1}^{n_{ob}} Q_i(x_c) \geq \tau$.

The locations of step 4 are precisely the locations where condition (C) is satisfied.

For a 30×30 reference grid, the size of $B_i = B_{y_i}$ needed to accommodate the range of scales and rotations mentioned above varies depending on the distance of the point y_i from the origin; it is at most on the order of 100 pixels. Note that B_y is entirely determined by y once the set of allowable transformations $\mathcal{A}$ is fixed.

Under condition (C) both p_o and p_b increase relative to their value on the reference grid. The new background probability p_b is approximated by $\lambda_b|B|$, where λ_b is the density of the features in background, and $|B|$ is the area of the regions B_i. The new object probability p_o can be reestimated from the training data using the regions B_i. The main concern is the increase in false positives due to the increase in the background probabilities. This is directly related to the background density λ_b. As long as λ_b is sufficiently small, this "greedy" detection algorithm will not yield too many false positives.

3.3 Local Features. The key question is which local features to use—namely, how to decrease λ_b while maintaining relatively high "on object" probabilities p_o. The first possibility would be to use the oriented edges, which serve as input to the entire system. These are the most basic local features detected in the visual system. The problem with edges, however, is

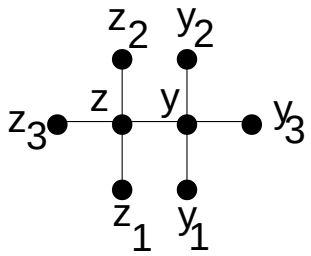

Figure 1: A vertical edge is present at z if $|I(z) - I(y)| > |I(z) - I(z_i)|$ and $|I(z) - I(y)| > |I(y) - I(y_i)|$ for all $i = 1, 2, 3$. The polarity depends on the sign of $I(z) - I(y)$. Horizontal edges are defined by rotating this scheme by 90 degrees.

that they appear with a rather high density in typical scenes. In other words λ_b is such that $\lambda_b|B| > 1$, leading to very large numbers of false positives. (See also the detailed discussion in Grimson, 1990.) The edge detector used here is a simple local maximum of the gradient with four orientations—horizontal and vertical of both polarities (see Figure 1). The edges exhibit strong invariance to photometric transformations and local geometric transformations. Typical λ_b for such edges on generic images is .03 per pixel.

We therefore move on to flexible edge conjunctions to reduce λ_b while maintaining high p_o. These conjunctions are flexible enough to be stable at specific locations on the object for the previously defined range of deformations, and they are lower density in the background. These conjunctions are simple binary filters computed on the edge maps. We emphasize that the computational approach and associated architecture can be implemented with other families of local features exhibiting similar statistical properties.

3.3.1 Flexible edge arrangements. Define a collection of small regions b_v in the immediate vicinity of the origin, indexed by their centers v. Let G_{loc} denote the collection of these centers. A local feature is defined in terms of n_{loc} pairs of edges and locations (e_ℓ, v_ℓ), $\ell = 1, \ldots, n_{loc}$. It is present at point x if there is an edge of type e_ℓ in the neighborhood $x + b_{v_\ell}$, for each $\ell = 1, \ldots, n_{loc}$. Figure 2 shows an example of an arrangement with three edges. It will be convenient to assume that $v_1 = 0$ and that $b_1 = \{0\}$. Invariance to local deformations is explicitly incorporated in the disjunction over the regions b_v. Note that each local feature can be viewed as a flexible star-type planar arrangement centered at x. Arrangements with two edges—$n_{loc} = 2$—loosely describe local features such as corners, t-junctions, slits, and various contour segments with different curvature.

On generic gray-scale images the density λ_b is found to decay exponentially with n_{loc} (see Amit & Geman, 1999), and for fixed n_{loc} is more or less the same no matter which arrangement is chosen. For example for $n_{loc} = 4$ the density λ_b lies between .002 and .003. For $n_{loc} = 2$, $\lambda_b \sim .007$.

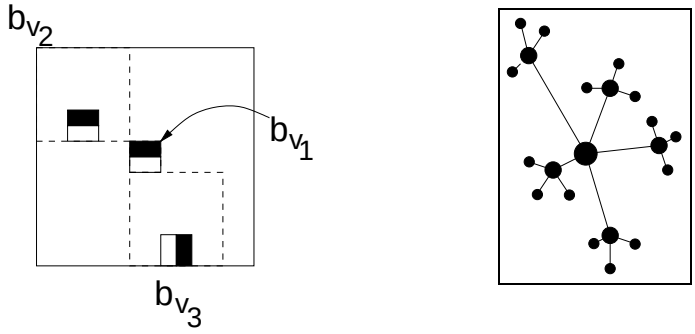

Figure 2: An arrangement with three edges. We are using $v_1 = (0,0), v_2 = (-2,2), v_3 = (2,-3)$. The regions b_{v_2} and b_{v_3} are 3×3, $b_1 = \{0\}$. This arrangement is present at a location x; if a horizontal edge is present at x, another horizontal edge is present anywhere in $x + b_{v_2}$ and a vertical edge is present anywhere in $x + b_{v_3}$. (Right) A graphical representation of the full model. The large circle in the middle represents the center point, each medium-sized circle represents the center of a local feature, and the small circles represent the edges

3.4 Training. Training images of the object class are registered to the reference grid. A choice of feature complexity is made; n_{loc} is chosen. At each location y a search through the pool of features with n_{loc} edges is carried out to find a feature that is present in over 50% of the data, anywhere in a small neighborhood of y. If one is found, it is recorded together with the location.

For a variety of object classes such as faces, synthetically deformed symbols and shapes, rigid 3D objects, and using $n_{loc} = 4$, a *greedy* search typically yields several tens of locations on the reference grid for which $p_o \sim .5$. After correcting for the larger regions B_i we obtain $p_o \sim 0.8$, and it turns out that taking into consideration the background statistics using only 20 of these feature and location pairs is sufficient. These are chosen randomly from all identified feature location pairs. We then use a threshold $\tau \sim 10$ in the Hough transform. Alternatively a *full* search over all features with two edges ($n_{loc} = 2$) yields $p_o > 0.9$. In this case 40 local features with $\tau \sim 30$ are needed for good classification. It is precisely the fact that this hierarchy of local features exhibits such statistics in real gray-level images that makes the algorithm feasible.

3.5 Some Experiments. In order to cover a range of scales of approximately 4:1, the same detection procedure is carried out at 5 to 6 resolutions. On a 200-Mhz Pentium-II the computation time for all scales together is about 1.5 seconds for generic 240×320 gray-scale images. Example detections are shown in Figures 7, 8, and 9. Also in these displays are random samples from the training sets and some of the local features used in each

representation in their respective locations on the reference grid. The detectors for the 2D view of the clip (see Figure 9) and for the "eight" (see Figure 8) were made from only one image, which was subsequently synthetically deformed using some random linear transformations. In the example of the clip, due to the invariances built into the detection algorithm, the clip can be detected in a moderate range of viewing angles around the original one, which were not present in training.

The face detector was produced from only 300 faces of the Olivetti database. Still faces are detected in very diverse lighting conditions and in the presence of occlusion. (More can be viewed online at http://galton.uchicago.edu/∼amit/detect.) For each image, we show both the detection of the Hough transform using a representation with 40 local features of two edges each. We also show the result of an algorithm that employs one additional stage of filtering of the false positives as described in Amit and Geman (1999). In this data set, there are on average 13 false positives per image at all six resolutions together. This amounts to .00005 false positives per pixel. The false-negative rate for this collection of images is 5%. Note that the lighting conditions of the faces are far more diverse than those on the Olivetti training set.

4 Neural Net Implementation

Our aim is to describe a network that can easily adjust to detect arrangements of local features representing different object classes. All features used in any object representation come from a predetermined pool $\mathcal{F} = \{\alpha_1, \ldots, \alpha_N\}$. The image locations of each feature in $\mathcal{F}$ are detected in arrays $F_i, i = 1, \ldots, N$ (see section 4.1.) A location x in F_i is **on** if local feature α_i is present at location x in the image.

Let $\mathcal{F} \times G$ denote the entire collection of pairs of features and reference grid locations—namely, $\mathcal{F} \times G = \{(\alpha_i, y), i = 1, \ldots, N,\ y \in G\}$. Define a module **M** that has one unit corresponding to each such pair—$N \times |G|$ units. An object representation consists of a small collection of n_{ob} such pairs $(\alpha_{i_j}, y_j), j = 1, \ldots, n_{ob}$. For simplicity assume that n_{ob} is the same for all objects and that τ is the same as well. Typically n_{ob} will be much smaller than N—on the order of several tens. *An object representation is a simple binary pattern in* **M**, *with* n_{ob} *1s and* $(N - n_{ob})$ *0s.*

For each local feature array F_i, introduce a system of arrays $Q_{i,y}$ for each location $y \in G$. These Q arrays lay the ground for the detection of *any* object representation by performing step 3 in the Hough transform, for each of the B_y regions. Thus a unit at location $x \in Q_{i,y}$ receives input from the region $x + B_y$ in F_i.

Note that for each unit $u = (\alpha_i, y)$ in **M**, there is a corresponding $Q_{i,y}$ array. All units in $Q_{i,y}$ receive input from u. This is where the top-down flow of information is achieved. A unit x in $Q_{i,y}$ is **on** only if both (α_i, y) is **on** and any unit in the region $x + B_y$ is **on** in F_i. In other words the representation

evoked in **M** *primes* the appropriate $Q_{i,y}$ arrays. The system of Q arrays sum into an array **S**. A unit at location $x \in \mathbf{S}$ receives input from all $Q_{i,y}$ arrays at location x and is **on** if $\sum_{i=1}^{N} \sum_{y \in G} Q_{i,y}(x) \geq \tau$. The array **S** therefore shows those locations for which condition (C) is satisfied.

If a unique location needs to be chosen, say, for the next saccade, then picking the x with the highest sum seems a natural choice. In Figures 7, 8, and 9, the red dots show all locations satisfying condition (C), and the green dot shows the one with the most hits. This architecture, called NETGLOB, is described in Figure 3.

For each local feature α_i we need $|G| = d^2$ complex Q-type arrays, so that the total number of Q-type arrays is $N_Q = Nd^2$. For example, if we use arrangements with two edges and assuming 4 edge types and 16 possible regions b_v (see section 3.3), the total number of features $N = 4 \times 4 \times 16 = 256$. Taking a 30×30 reference grid N_Q is on the order of 10^5. It is, of course, possible to give up some degree of accuracy in the locations of the y_j's, assuming, for example, they are on some subgrid of the reference grid. Recall that the **S** array is only supposed to provide candidate locations, which must then undergo further processing after foveation, including small corrections for locations. Therefore, assuming, for example, that the coordinates of y_j are multiples of three would lead to harmless errors. But then the number of possible locations in the reference grid reduces to 100, and $N_Q \sim 10^4$. Using such a coarse subgrid of G also allows the F and Q arrays to have lower resolution (by the same factor of three), thus reducing the space required by a factor of nine. Nonetheless, N_Q is still very large due to the assumption that all possible local features are detected bottom-up in hard-wired arrays F_i. This is quite wasteful in space and computational resources. In the next section we introduce adaptable local feature detectors, which greatly reduce the number of Q-type layers needed in the system.

4.1 Detecting the Local Features. The local features are local star-type arrangements of edges; they are detected in the F_i layers with input from edge detector arrays E_e, $e = 1, \ldots, 4$. For each edge type e, define a system of "complex" arrays $C_{e,v}$, $v \in G_{loc}$. A unit $x \in C_{e,v}$ receives input from the region $x + b_v$ in E_e. It is **on** if any unit in $x + b_v$ is **on** . A local feature α_i with pattern (e_ℓ, v_ℓ), $\ell = 1, \ldots, n_{loc}$, as defined in section 3.3, is detected in the array F_i. Each $x \in F_i$ receives input from $C_{e_\ell, v_\ell}(x)$, $\ell = 1, \ldots, n_{loc}$ and is **on** if all n_{loc} units are **on** . The full system of E, C, and F layers, called NETLOC, is shown in Figure 4.

Note that in contrast to the global detection scheme, where a top-down model determined which arrays contributed to the summation array **S**, here everything is hard-wired and calculated bottom-up. Each unit in a F_i array sums the appropriate inputs from the edge detection arrays.

Alongside the hard-wired local feature detectors we introduce *adaptable* local feature detectors. Here local feature detection is driven by top-down

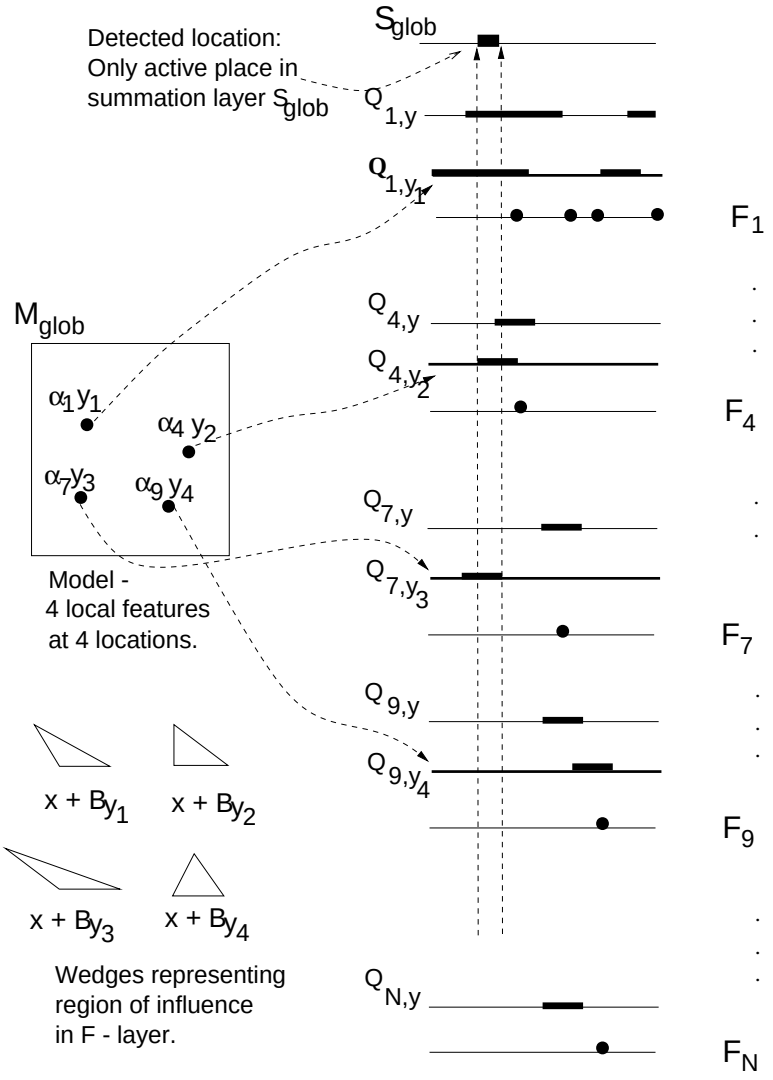

Figure 3: NETGLOB. The object is defined in terms of four features at four locations. Each feature/location (α_i, y) pair provides input to all units in the corresponding $Q_{i,y}$ array (thick lines). The locations of the feature detections are shown as dots. They provide input to regions in the Q arrays shown as double-thick lines. At least three local features need to be found for a candidate detection. The fourth feature does not contribute to the detection; it is not present in the correct location. There are N systems of F, Q arrays—one for each $\alpha_i \in \mathcal{F}$.

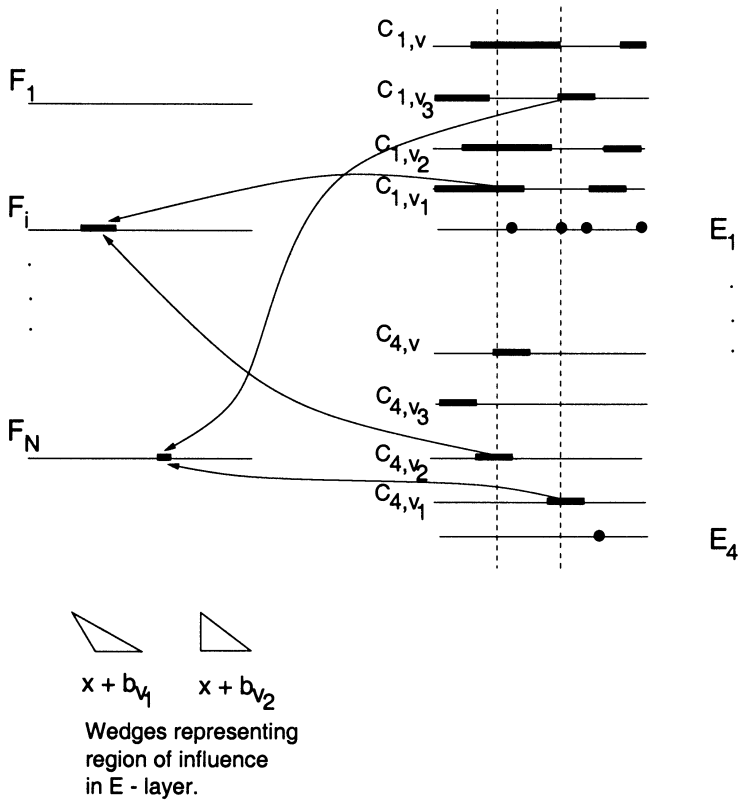

Figure 4: NETLOC. Four E layers detect edge locations. For each E_e layer, there is a $C_{e,v}$ corresponding to each $v \in G_{loc}$—all in all, $4 \times |G_{loc}| - C$ layers. There is an F layer for each local feature, each location in the F layer receives input from the same location in C_{e_ℓ,v_ℓ}, $\ell = 1, \ldots, n_{loc}$.

information. Only features required by the global model are detected. If only adaptable features detectors are used in the full system, the number of Q arrays required for the system is greatly reduced and is equal to the size of the reference grid.

For each location y in the reference grid, there is a module $M_{loc,y}$, a detector array $F_{loc,y}$, one "complex" array Q_y, and a dedicated collection of $C_{e,v}$ arrays. Without a local feature representation being evoked in $M_{loc,y}$ nothing happens in this system. When a local feature representation is turned on in $M_{loc,y}$, only the corresponding $C_{e,v}$ arrays are primed. Together with the input from the edge arrays, the local feature locations are detected in a summation array array $F_{loc,y}$. Finally a detection at x in $F_{loc,y}$ is spread to the region $x - B_y$ in Q_y. (See Figure 5.)

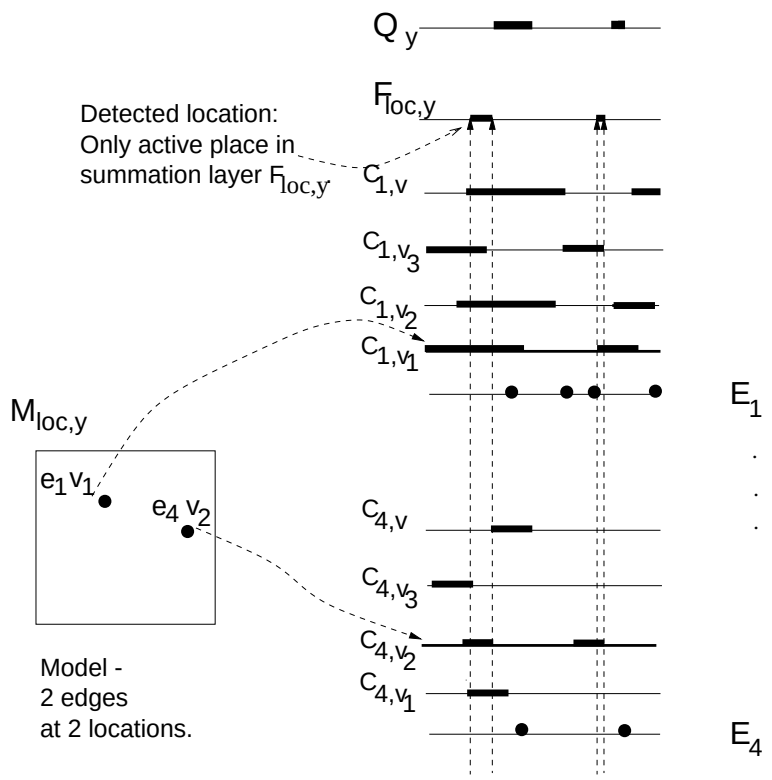

Figure 5: NETADAP. The representation of the local feature is turned on in the local feature memory module. This determines which C layers receive input from $M_{loc,y}$. Units in these C layers that also receive input from the edge layer become activated. The $F_{loc,y}$ layer sums the activities in the C layers and indicates the locations of the local features. $F_{loc,y}$ then feeds into Q_y: $x \in Q_y$ is **on** if any point in $x + B_y$ is on in $F_{loc,y}$.

A global system with only adaptable feature detections is shown in Figure 6. In this system the number of Q type arrays is $|G|$. Top-down information does not go directly to these Q layers. When an object representation is turned on in **M**, each unit (α_i, y) that is **on** activates the representation for α_i in $M_{loc,y}$, which primes the associated $C_{e,v}$ arrays. The detections of these features are obtained in the $F_{loc,y}$ arrays and appropriately spread to the Q_y arrays. All the arrays Q_y are summed and thresholded in **S** to find the candidate locations.

The adaptable local feature detectors $F_{loc,y}$ can operate alongside the hard-wired F arrays. The arrays Q_y, together with the original system of

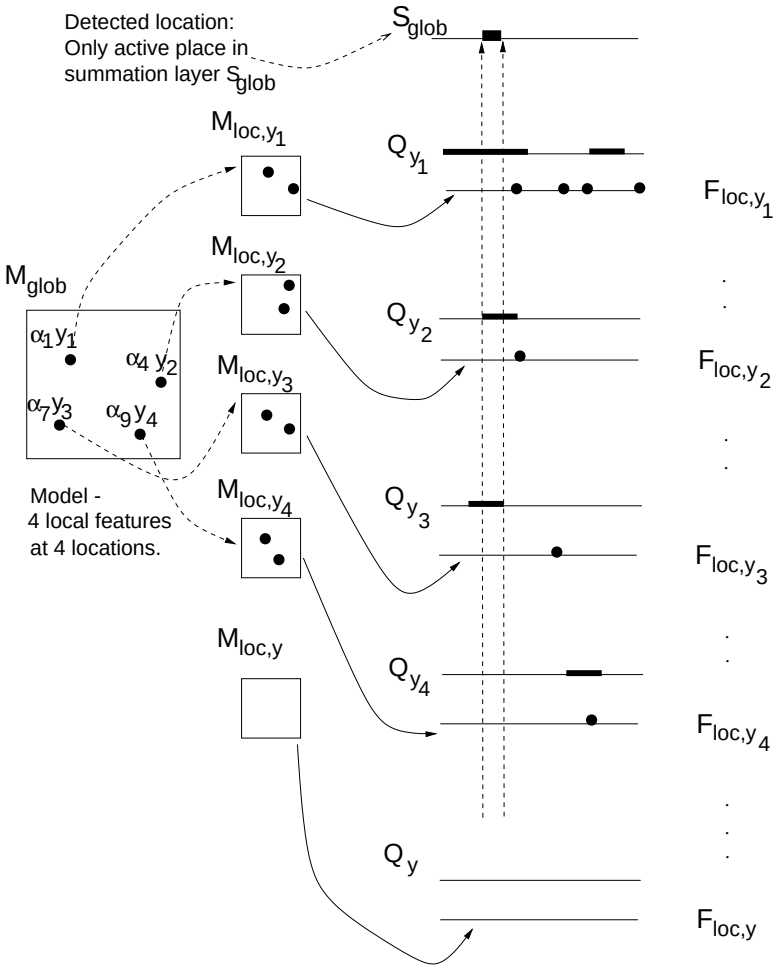

Figure 6: NETGLOB (adaptable). The object is defined in terms of four features at four locations. Each location y provides input to an $M_{loc,y}$. If a pair (α, y) is **on** in M_{glob}, the representation of α is **on** in $M_{loc,y}$. This is represented by the dots in the corresponding modules. The detections of feature α are then obtained in $F_{loc,y}$, using a system as in Figure 5, and spread to Q_y. At least three local features need to be found for a candidate detection. The fourth feature does not contribute to the detection; it is not present in the correct location. There was no feature **on** at location y, so the system of $M_{loc,y}$ is not active.

Figure 7: Each of the top two panels shows 10 of the 20 local features from the face representation, trained from 300 faces of the Olivetti database. The blue-green combinations indicate which edge is involved in the edge arrangement. Blue stands for darker. The three pink dots are the locations of the two eyes and mouth on the reference grid. Each red dot in an image represents a candidate location for the center of the face. The green dot shows the location with most counts. Detections are shown from all six resolutions together. The top image was scanned from *Nature*, vol. 278, October 1997.

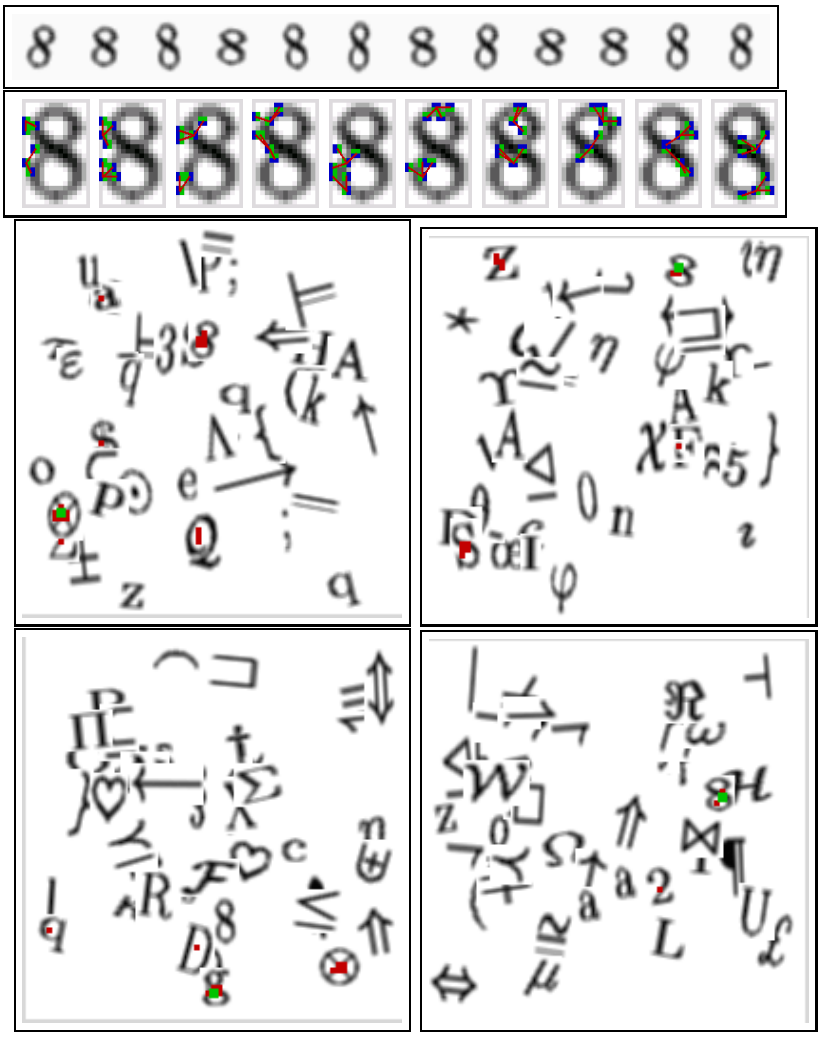

Figure 8: First panel: 12 samples from the training set. Second panel: 10 images of the prototype 8 with two different local features in each. Detections are shown in four randomly generated displays; red dots indicate detected locations, and the green dot indicates the location with the highest count.

$Q_{i,y}$ arrays, sum into S_{glob}. If a local feature (α_i, y) is **on** in **M**, either there is an associated and hard-wired detector layer F_i and $Q_{i,y}$ is primed, or it feeds into $M_{loc,y}$ and the required detections are obtained in $F_{loc,y}$ and spread to the associated Q_y. It appears more reasonable to have a small number of re-

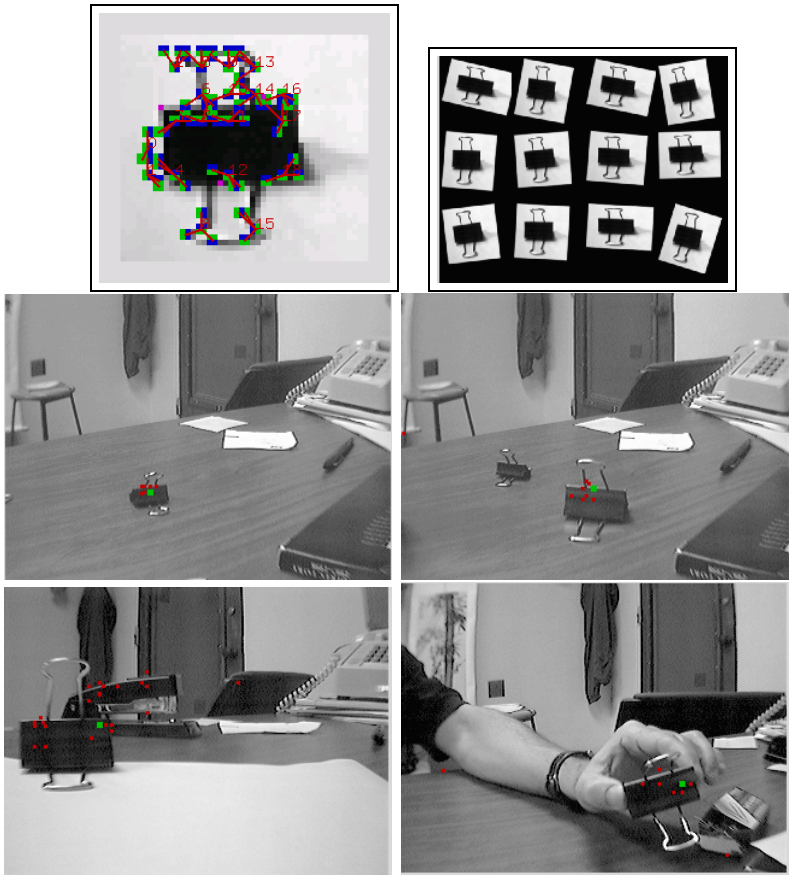

Figure 9: Same information for the clip. The clip was trained on 32 randomly perturbed images of one original image, 12 of which are shown. Detection is only for this range of views, not for all possible 3D views of the clip.

curring local features that are hard-wired in the system alongside adaptable local feature detectors.

5 Biological Analogies

5.1 Labeling the Layers. Analogies can be drawn between the layers defined in the architecture of NETGLOB and existing layers of the visual system. The simple edge-detector layers E_e, $e = 1 \ldots, 4$ correspond to simple orientation selective cells in V1. These layers represent a schematic abstraction of the information provided by the biological cells. However, the

empirical success of the algorithm on real images indicates that not much more information is needed for the task of selection.

The $C_{e,y}$ layers correspond to complex orientation selective cells. One could imagine that all cells corresponding to a fixed-edge type e and fixed image location x are arranged in a column. The only difference as one proceeds down the column is the region over which disjunction (ORing) is performed (e.g., the receptive field). In other words, the units in a column are indexed by $v \in G_{loc}$, which determines the displaced region b_v. Indeed, any report on vertical electrode probings in V1, in which orientation selectivity is constant, will show a variation in the displacement of the receptive fields. (See Zeki, 1993.)

The detection arrays F_i correspond to cells in V2 that respond to more complex structures, including what is termed as illusory contours (see von der Heydt, 1995). If an illusory contour is viewed as a loose local arrangement of edge elements, that is precisely what the F_i layers are detecting. The $Q_{i,y}$ layers correspond to V4 cells. These have much larger receptive fields, and the variability of the location of these fields as one proceeds down a column is much more pronounced than in V1 or V2; it is on the order of the size of the reference grid.

5.2 Bottom-Up–Top-Down Interactions. Top-down and bottom-up processing are explicitly modeled. Bottom-up processing is constantly occurring in the simple cell arrays E_e, which feed into the complex cell arrays $C_{e,y}$ which in turn feed into the F_i–V2 type arrays. One could even imagine the Q–V4 type arrays being activated by the F arrays regardless of the input from **M**. Using the terminology introduced in Ullman (1996), only those units that are primed by receiving input from **M** contribute to the summation into **S**.

The object pattern that is excited in the main memory module determines which of the Q arrays will have enhanced activity toward their summation into **S**. Thus the final determination of the candidate locations is given by an interaction of the bottom-up processing and the top-down processing manifested in the array **S**. With the adaptable local feature detectors, the top-down influence is even greater. Nothing occurs in an $F_{loc,y}$ detector unless a local feature representation is turned on in $M_{loc,y}$ due to the activation in **M** of an object representation that contains that feature at location y.

5.3 Gating and Invariant Detection. The summation array **S** can serve as a gating mechanism for visual attention. Introduce feedback connections from each unit $x \in \mathbf{S}$ to the unit at the same location in all Q layers. The only way a unit in a $Q_{i,y}$ layer can remain active for an extended period is if it received input from some $(\alpha_i, y) \in \mathbf{M}$ *and* from the **S** layer.

This could provide a model for the behavior of IT neurons in delay match-to-sample task experiments, (see Chelazzi, Miller, Duncan, & Desimone, 1993, and Desimone et al., 1995). In these experiments neurons in IT se-

lectively responsive to two different objects are found. The subject is then presented with the object to be detected: the sample. After a delay period, an image with both objects is displayed, both displaced from the center. The subject is supposed to saccade to the sample object. After presentation of the test image, neurons responsive to *both* objects are active. After about 100 milliseconds and a few tens of milliseconds prior to the saccade, the activity of the neurons selective for the nonsample object decays. Only neurons selective for the sample object remain active.

If *all* units in the $Q_{i,y}$ layer feed into (α_i, y) in **M**, then (α_i, y) receives bottom-up input whenever α_i is present *anywhere* in the image, and no matter what y is. Thus, the units in **M** do not discriminate between the locations from which the inputs are coming, and at the presentation of a test image, one may observe activity in **M** due to bottom-up processing in the Q layers, which responds to other objects in the scene. This is consistent with scale and location invariance observed in IT neuron responses (Ito et al., 1995). When a location is detected in **S**, gating occurs as described above, and the only input into **M** from the Q arrays is from the detected location *at the layers corresponding to the model representation*.

For example assume that feature α_i is detected at locations x_1, x_2, x_3 in the image (i.e., units x_1, x_2, x_3 are **on** in F_i). Then all the units in $x_1 - B_y$, $x_2 - B_y$, $x_3 - B_y$ will be **on** in $Q_{i,y}$ for all $y \in G$. This corresponds to the bottom-up flow of information. In the object model, assume α_i is paired with location $\hat{y}$. Assume that only x_1 comes from the correct location on an instance of the object, that is, the center of the object lies around $x_1 - B_{\hat{y}}$. Then if enough other features are detected on this instance of the object, some point $x^* \in x_1 - B_{\hat{y}}$ will be **on** in **S**, and subsequently the area around x^* in the $Q_{\hat{y},i}$ layer will remain active.

After detection occurs in **S** followed by the gating process, the only input coming into **M** from the Q arrays corresponding to feature α_i $(Q_{i,y}, y \in G)$ is from $Q_{i,\hat{y}}(x^*)$. The same is true for the other features in the object representation that have been detected at location x^*. This means that after detection, the **M** module is either receiving no input (if no candidate locations are found) or some subset of the units in the object representation is receiving reinforced input. *Only* the object representation is reinforced in **M**, signifying that detection has occurred invariant to translation scale and other deformations.

To summarize, the units in **M** exhibit selective activity for the sample object, only due to the input from a lower-level process in which location has already been detected and gating has occurred.

5.4 Hard-Wired Versus Adaptable Features. The number of hard-wired local feature arrays F_i and $Q_{y,i}$ we would expect to find in the system is limited. It is reasonable to assume that a complementary system of adaptable features exists as described in section 4.1. These features are learned for more specific tasks and may then be discarded if not needed; for example,

finer angle tuning of the edges may be required, as in the experiments in Ahissar and Hochstein (1997). People show dramatic improvement in detection tasks over hundreds of trials in which they are expected to repeat the task. It may well be that the hard-wired local features are too coarse for these tasks. The repetition is needed to learn the appropriate local features, which are then detected through the adaptable local feature system.

5.5 Learning. Consider the reference grid as the central field of view (fovea), and let each unit $(\alpha_i, y) \in \mathbf{M}$ be directly connected to $F_i(y)$ for the locations $y \in G$. The unit $(\alpha_i, y) \in \mathbf{M}$ is **on** if $F_i(y)$ is on. In other words, the area of the reference grid in the F_i arrays feeds directly into the **M** module.

Learning is then done through images of the object class presented at the reference grid. Each such image generates a binary pattern in the module **M**. In the current implementation (see section 3.4), a search is carried out at each location $y \in G$ to identify stable features for each location. These are then *externally* stored.

However, it also possible to consider **M** as a connected neural net that employs Hebbian learning (see, for example, Brunel, Carusi, & Fusi, 1998, and Amit & Fusi, 1994) to store the binary patterns of high-frequency units for an object class *internally*. These patterns of high-frequency units would be the attractors generated by presentations of random samples from the object class. The attractor pattern produced in **M** then activates the appropriate Q layers for selection, as described above.

6 Discussion

The architecture presented here provides a model for visual selection when we know what we are looking for. What if the system is not guided toward a specific object class? Clearly high-level contextual information can restrict the number of relevant object classes to be entertained (see Ullman, 1996). However, even in the absence of such information, visual selection occurs. One possibility is that there is a small collection of generic representations that are either learned or hard-wired, which are useful for "generic" object and background discrimination and have dedicated systems for detecting them. For example, we noticed, not surprisingly, that the object representation produced for the "clip" (see Figure 9) often detects generic rectangular, elongated dark areas.

Many psychophysical experiments on detection and visual selection involve measuring time as a function of the number of distractors. In Elder and Zucker (1993) there is evidence that detection of shapes with "closure" properties does not slow linearly with the number of distractors, whereas shapes that are "open" do exhibit slowing; Also in Treisman and Sharon (1990) some detection problems exhibit slowing; others do not. This is directly related to the issue of false positives. The architecture presented above is parallel and is not affected by the number of distractors. However, the number of

distractors can influence the number of false positives. The chance that the first saccade moves to the correct location decreases, and slowing down occurs. How can the effect of the distractors be avoided? Again, this could be achieved using generic object representations with hard-wired detection mechanisms, that is, a summation layer **S** dedicated to a specific representation, and hence responding in a totally bottom-up fashion.

This discussion also relates to the question of how the representation for detection interacts with representations for more detailed recognition and classification, which occur after foveation. How many false positives can detection tolerate, leaving final determination to the stage after foveation?

There are numerous feedback connections in the visual system. In the current architecture, there is no feedback to the local feature detectors and edge detectors beyond the basic top-down information provided by the model. Also the issue of larger rotations has not been addressed. The local features and edges are not invariant to large rotations. How to deal with this issue remains an open question. Finally, in the context of elementary local cues, we have not made use of anything other than spatial discontinuities: edges. However, the same architecture can incorporate local motion cues, color cues, depth cues, texture, and so forth. Local features involving conjunctions of all of these local cues can be used.

Acknowledgments

I am indebted to Daniel Amit and Donald Geman for numerous conversations and suggestions. The experiments in this article would not have been possible without the coding assistance of Keneth Wilder. This work was supported in part by the Army Research Office under grant DAAH04-96-1-0061 and MURI grant DAAHO4-96-1-0445.

References

Ahissar, M., & Hochstein, S. (1997). Task difficulty and visual hierarchy: Reverse hierarchies in sensory processing and perceptual learning. *Nature*, *387*, 401–406.

Amit, Y. (1998). *Deformable template methods for object detection* (Tech. Rep.). Chicago: Department of Statistics, University of Chicago.

Amit, D. J., & Fusi, S. (1994). Dynamic learning in neural networks with material synapses. *Neural Computation*, *6*, 957.

Amit, Y., & Geman, D. (1999). A computational model for visual selection. *Neural Computation*, *11*, 1691–1715.

Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*, 111–122.

Brunel, N., Carusi, F., & Fusi, S. (1998). Slow stochastic Hebbian learning of classes of stimuli in a recurrent neural network. *Network*, *9*, 123–152.

Burl, M. C., Leung, T. K., & Perona, P. (1995). Face localization via shape statistics. In M. Bichsel (Ed.), *Proc. Intl. Workshop on Automatic Face and Gesture Recognition* (pp. 154–159).

Chelazzi, L., Miller, E. K., Duncan, J., & Desimone, R. (1993). A neural basis for visual search in inferior temporal cortex. *Nature*, *363*, 345–347.

Cootes, T. F., & Taylor, C. J. (1996). Locating faces using statistical feature detectors. In *Proc., Second Intl. Workshop on Automatic Face and Gesture Recognition* (pp. 204–210).

Desimone, R., Miller, E. K., Chelazzi, L., & Lueschow, A. (1995). Multiple memory systems in visual cortex. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 475–486). Cambridge, MA: MIT Press.

Elder, J., & Zucker, S. W. (1993). The effect of contour closure on the rapid discrimination of two-dimensional shapes. *Vison Research*, *33*, 981–991.

Fukushima, K. (1986). A neural network model for selective attention in visual pattern recognition. *Biol. Cyber.*, *55*, 5–15.

Fukushima, K., & Wake, N. (1991). Handwritten alphanumeric character recognition by the neocognitron. *IEEE Trans. Neural Networks*, *2*, 355–365.

Grimson, W. E. L. (1990). *Object recognition by computer: The role of geometric constraints*. Cambridge, MA: MIT Press.

Ito, M., Tamura, H., Fujita, I., & Tanaka, K. (1995). Size and position invariance of neuronal response in monkey inferotemporal cortex. *Journal of Neuroscience*, *73*(1), 218–226.

Lueschow, A., Miller, E. K., & Desimone, R. (1994). Inferior temporal mechanisms for invariant object recognition. *Cerebral Cortex*, *5*, 523–531.

Rojer, A. S., & Schwartz, E. L. (1992). A quotient space Hough transform for space-variant visual attention. In G. A. Carpenter & S. Grossberg (Eds.), *Neural networks for vision and image processing*. Cambridge, MA: MIT Press.

Rowley, H. A., Baluja, S., & Takeo, K. (1998). Neural network-based face detection. *IEEE Trans. PAMI*, *20*, 23–38.

Sung, K. K., & Poggio, T. (1998). Example-based learning for view-based face detection. *IEEE Trans. PAMI*, *20*, 39–51.

Thorpe, S., Fize, D., & Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, *381*, 520–522.

Treisman, A., & Sharon, S. (1990). Conjunction search revisited. *Journal of Experimental Psychology: Human Perception and Performance*, *16*, 459–478.

Ullman, S. (1996). *High-level vision*. Cambridge, MA: MIT Press.

Van Rullen, R., Gautrais, J., Delorme, A., & Thorpe, S. (1998). Face processing using one spike per neuron. *Biosystems*, *48*, 229–239.

von der Heydt, R. (1995), Form analysis in visual cortex. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 365–382). Cambridge, MA: MIT Press.

Wiskott, L., Fellous, J.-M., Kruger, N., & von der Marlsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Trans. on Patt. Anal. and Mach. Intel.*, *7*, 775–779.

Zeki, S. (1993). *A vision of the brain*. Oxford: Blackwell Scientific Publications.

Received November 19, 1998; accepted July 12, 1999.