Computer Science Department Technical Report
Machine Perception Laboratory
University of California
Los Angeles, CA 90024-1596

A NEURAL ARCHITECTURE FOR TEXTURAL SEGMENTATION

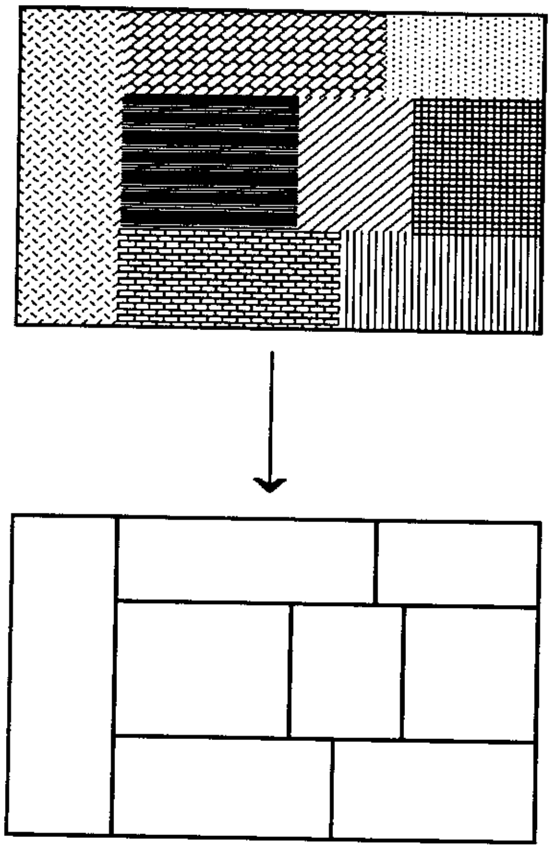
Edmond Mesrobian
Josef Skrzypek

December 1989
CSD-890068

MPL

Machine
Perception
Lab

UCLA
Computer Science
Department



A Neural Architecture for Textural Segmentation

Edmond Mesrobian
Josef Skrzypek

TR 89-4

MICRO-Hughes Grant
#541122-57442

MPL

Machine
Perception
Lab

top-down

bottom-up

A Neural Network Architecture for Textural Segmentation

Edmond Mesrobian Josef Skrzypek

Machine Perception Laboratory
Computer Science Department
University of California
Los Angeles, CA 90024

Abstract

In this paper we examine some parallel architectures designed for image processing and then address their applicability to the problem of textural segmentation. Using this information and research into the structure of the human visual system, an abstract neural network architecture for textural segmentation is proposed. The underlying premise is that textural segmentation can be improved by combining information from multiple sources. Large scale image features such as intensity help guide the analysis of local differences in texture elements (texels) to extract better boundaries. This approach differs from most of the previous work where the differences in global, second-order statistics of the image points are used as the basis for segmentation. A possible parallel implementation of this abstract neural network architecture consists of a hierarchy of functionally different "neurons". In the lower layers of this architecture, simple features are extracted from the image. These simple features are linked together to form more complex texels. Finally, local and more global differences in texels or their organization are enhanced and linked into boundaries.

Introduction

One of the most difficult problems in vision is segmentation, an operation performed by middle-level visual functions. Here, regions of the image are delineated based on similarities in pixel characteristics or other local/global measures derived by either using bottom-up approaches to data processing and/or top-down strategies using prior knowledge from higher-level cognitive processes. Eventually, the result of this aggregation leads to the labeling of objects and the symbolic description of relationships among them. The process of texture synthesis and analysis is central to segmentation. Various techniques have been used to achieve segmentation, for example, region growing, edge detection and linking, piecewise approximation, clustering and many others [1]. However, none of these techniques can guarantee the generation of segments that truly represent the perceived objects. The solution to this problem in general requires one to understand what vision is. Segmentation must be based on a rich description of the image. This can be achieved if systems are equipped to process high-resolution input data into viewpoint and ambient light independent images, segmented according to edges, texture, color, motion, depth and shape. Our approach to solving segmentation encompasses all attributes but in this paper we will focus only on texture. The underlying philosophy is that algorithms and architectures for low-level vision are most directly influenced by knowledge derived from the neurosciences. This is perhaps the first area where the close exchange of ideas will benefit both fields.

What is texture? Texture can be defined as a local relation repeated over a region much larger than the spatial size of the relation [2]. The relation is a non-random arrangement of elementary and regular parts/attributes/primitives, with the primitives being uniform everywhere in the textured region. Equivalently, texture can be characterized by a two level description [3]. At the microscopic level texture is described by grain elements, while at a macroscopic level it is patterns or structures formed from the regular or random repetition of the grain elements.

Although most approaches to textural segmentation concentrate on defining texture in global terms using statistical methods to detect texture elements (texels) and to derive structural relationships between the texels, these approaches have not resulted in a solution for textural segmentation (for a review see [4]). Purely statistical methods include: measures such as entropy, contrast, and correlation based on grey tone co-occurrence matrices [5]; measures of texture edgeness and energy [6]; and generalized co-occurrence matrices [7]. The major problem with these statistical methods lies in the fact that not all textures are completely definable by mathematical formulations. Most of these measures seem to be unrelated to current knowledge about the neurophysiology of vision. Finally, psychophysical evidence strongly suggests that humans do not employ second- or higher-order statistical measures for texture perception [8,9].

Statistical measures were also used by Vilnrotter, et. al. in their structural approach to texture analysis [10]. They gathered co-occurrence statistics on edge data at various orientations and distances. Size and spacing information extracted from these co-occurrence measures were used to detect primitive texels. Next, centroids were computed for each of the primitive texels found in the image, and placement rules were derived to provide spatial relationships between the centroids of the different primitive texels. The texture pattern was then reconstructed by placing texels according to the placement rules. Although this approach performs reasonably well on a variety of textures, its algorithmic nature does not seem to be related to human texture perception.

Our approach is a derivative of previous attempts at textural segmentation initiated by Marr [11], Beck [12], Julesz [8], and Wermser [13]. Marr's computational theory of texture vision rests upon grouping functions and first-order discriminations performed on the primal sketch of the image. The grouping functions include curvilinear aggregation (local orientation information is used to produce contours by joining nearby, aligned tokens) and theta aggregation (grouping of local, similarly oriented tokens in a direction which differs from their intrinsic orientation). First-order discriminations are based on easy to compute global measures such as the total number of elements at each orientation and the total contour length at each orientation.

The theory of textural segmentation proposed by Beck, et. al., is an algorithm consisting of four steps. First, simple features are extracted from the image. Local operations are then employed to link these features to form higher-level textural elements. Next, operators encode differences in features in neighboring regions of various sizes. Finally, decision units segment the scene by analyzing the magnitude, number, and distribution of these difference signals. However, Beck has recently argued that tripartite (a three part texture pattern) segregation is primarily a function of spatial frequency components and not local grouping processes [14]. This suggests that higher-order processes in texture segmentation have access to information corresponding to the outputs of spatial frequency channels.

Julesz originally proposed that humans are unable to discriminate between patterns with identical first- and second-order statistics [15]. Since many counterexamples were found to disprove this conjecture [9], his new model [8] resembles the model proposed by Beck; first-order differences in local features provide the basis for preattentive texture discrimination. These elements of texture perception (textons) can be grouped into three classes: color, elongated blobs (of a given orientation, width, and aspect ratio), and terminators (of blobs).

An attempt to implement the concept of local interaction between texels was made by Wermser. His model for textural analysis consists of two modules. One module computes first-order statistics, mean and variance, on a $\log I$ image (logarithmic transformation of the original intensity image) passed through a Modulation Transfer Function (MTF) bandpass filter. The other module locates edge activity using four directional operators at three spatial scales. With four directional operators at three different scales and the mean and variance, Wermser obtains a 14 element feature vector for every pixel in the image. This information is then used to segment the image.

In this paper, we propose a solution to *textural segmentation* in the form of a *3D connectionist structure* rather than an algorithm which produces feature vectors describing textures. Our approach is inspired by clues for a solution found in the neurophysiology of natural vision [16,17]. The underlying premise of our approach is that textural segmentation can be achieved by recognizing local differences in texels. The recognition of local differences can be made more robust by utilizing large scale image features such as intensity or color. The architecture consists of a hierarchy of functionally different "neurons" (nodes). Elementary feature detectors are initially applied to the image to extract primitive texels. Second, Gestalt rules are used to group these simple features into uniformly textured regions. The smallest uniform region over which a Gestalt organization emerges is defined recursively as a higher-order texel. Finally, local and more global differences in texels or their organization are enhanced and linked into boundaries.

Motivation for a Parallel Computing Architecture

Our parallel computing architecture is motivated by the need for vision systems capable of real-time performance [18]. This implies computing speeds in excess of 1 to 100 billion operations per second, which is a difficult demand for a traditional single instruction stream, single data stream (SISD) architecture. Since the image data is inherently parallel, most of the low-level vision processing tasks are often similar over the spatial extent of the image. Hence, one possible solution is to develop a parallel computing architecture for low-level vision. For example, detecting edges (intensity discontinuities) can be a computationally intensive task on a traditional Von Neumann machine. To convolve an image with a 3×3 mask requires 9 multiplications and 8 additions at each image pixel. An alternative is to perform the entire operation simultaneously by using an array of dedicated convolvers, thus significantly reducing the computation time.

Additional support for the development of a highly parallel computing architecture is supplied by neurophysiological data. Hubel and Weisel discovered parallel computing structures in the visual cortex [16]. They found vertical slabs of neural tissue composed of neurons overlooking a particular portion of the visual field. Some of these neurons are sensitive to contrast changes at certain preset orientations, widths, and sizes, while others are sensitive to color, motion and binocular disparity. Single neurons in the cat striate cortex were found to be sensitive to differences in texture luminance [19]. Structural (orientational) differences in texture are perhaps encoded by a different set of neurons. Differences in both texture luminance and texture orientation, collectively providing a perceptual impression of a texture border, are processed by neurons at an even higher layer. Our architecture is consistent with these findings.

Existing Architectures for Image Processing

The focus of this paper is on parallel architectures, implemented or conceptual, which are particularly well suited for texture discrimination. Many parallel architectures have been designed and implemented (for a review see [20]). These include pyramid architectures [21], systolic arrays [22,23], SIMD and MIMD hypercube architectures [24,25], and SIMD mesh connected architectures [26]. These architectures run the gamut from fine grain to coarse grain processing units. Typically, fine grain processors are ideal for highly structured tasks such as convolving an image with a filter, while coarse grain processors provide computational power for highly symbolic processing tasks.

Pyramid Architectures

Pyramids are data structures which provide successively condensed representations of the information in the input image, but at the same time retain the important features of the lower level representations in topological order [27,28]. For example, a pyramid starts at the lowest level with an input image of size $2^n \times 2^n$, and ends at the top with dimension 1×1. Each node on a particular level obtains its information as a result of computations based on its four children nodes, or possibly some larger neighborhood. A node k levels above the pyramid's base ultimately receives information from a $2^k \times 2^k$ portion of the image in the base. By assigning a processor to each node of the pyramid, some important image processing computations can be performed. VLSI chip layouts for pyramid computers have been proposed by many researchers [29,30]. Dyer's [30] pyramid computer of base size n^2 is a 4-ary processor tree of depth $\log n$, with each processor connected to its father, 4 brothers, and 4 sons.

As a simple example, a pyramid can be used to repeatedly average the pixel intensities in the image. As one proceeds up the pyramid, a node at a particular level represents the average of the four nodes below it. At the apex of the pyramid is the average intensity of the entire image. However, pyramid nodes do not necessarily have to represent pixel intensities. They can also represent texture properties [27]. This enables the computation of textural properties such as texture density and orientation.

Another example of a global operation is a Gaussian pyramid [3], where each successively higher level is a low-pass filtered version of the previous level. Hierarchical Discrete Correlation (HDC) is another global operation that can be performed using a pyramid [3]. It is similar to the Gaussian pyramid, except that the distance between the nodes that contribute to the calculation doubles with each iteration as one progresses up the pyramid. One approach to texture feature extraction using these functions is to compute a Laplacian pyramid for a given image, square it, then apply the HDC function to each of the Laplacian layers. This is analogous to estimating the local power spectrum for various resolutions. The HDC function acts to integrate the power measurements detected by the squared Laplacian function. Once the raw texture information is obtained from various size channels the interesting problem becomes how to use it for textural segmentation.

Topological preservation is an important feature of pyramids. One of the biggest problems with attempting computer recognition of texture is to determine the scale at which to have the computer's vision system view an image. Human perception of texture does not change with viewing distance [3] perhaps because of size constancy or because of an auto-ranging window selector. Since a pyramid preserves image topology, it is possible to have different hierarchical layers represent different windows. A texture that is too small to be detected by one level of the pyramid can be recognized by a lower (finer) level.

Systolic Arrays

One limitation of most pyramid architectures is the absence of lateral connections between elements of the same hierarchical level. This problem is solved in systolic arrays where data is passed from one processing element to the next without being stored back in memory. Because of this feature, systolic arrays can achieve a high throughput in spite of a relatively low memory bandwidth [31].

Convolution, correlation, and filtering, three image processing functions particularly applicable to texture parameterization, can be easily implemented on this hardware. Kung and Song [22] designed a 2D systolic chip which has a hierarchy of k $k \times k$ (k is the size of the convolution window) mesh connected processor arrays for 2D convolution. The convolution is performed by pipelining the image data through the k arrays.

An especially attractive concept is a programmable systolic array, where the function of the array can be programmed rather than selected from a limited set of algorithms. Interconnections between the array elements may also be programmable at the cost of performance.

A suitably designed array could rapidly compute primitive texture elements for an image. The problem lies in attempting to use the systolic array for the next step — combining the texture primitives into either higher-level texture elements, or for segmenting the image. Higher-level processing requires the ability to look at larger regions of the image in order to link low-level features. This implies that the systolic array must be large enough to handle the larger amount of data efficiently. For portions of the computation, some of the power of the systolic array will be idle. Since systolic arrays compute new values as a moving front, staging the data properly so the right datum arrives at the correct input of an array cell at the proper time is a difficult and critical problem. Ideally we would like to have local computations performed simultaneously throughout the array.

SIMD and MIMD Hybercube Architectures

In a hypercube architecture with n processors, each processing processing element is physically connected to $\log n$ neighboring processors and communicates with any other processor in $O(\log n)$. This design has been employed to develop architectures based on SIMD and MIMD processors.

The Connection Machine (CM-1) [24] is a SIMD hypercube architecture which consists of 64K bit-serial processors each with 4K bits of memory (also accessible by all of the processors). A processor can send messages directly to any of the 4 processors to which it is physically connected. Message passing between processors that are not physically connected is handled via message routers. In the current version of the Connection Machine CM-2, each processor has 64K bits of memory and shares a floating point processor with its 16 neighboring processors. Programming the CM-2 is simplified through the use of high-level languages *Lisp and C*. The CM-2 has been used to implement a variety of vision algorithms such as edge detection, histogramming, Hough transforms, and connected component labeling [32].

The Intel iPSC hypercube architecture [25] provides a flexible interconnection of 128 MIMD processing units, each with 1 Mbyte of memory. A processor is physically connected to only eight other processors. Therefore, to communicate with any of the remaining processors requires passing messages through intermediate processors acting as relay stations.

SIMD Mesh Connected Architectures

These highly parallel single instruction, multiple data architectures permit fast computation of some basic, and other more sophisticated image processing functions. Two dimensional mesh connected architectures consist of a large number of processing elements arranged in a square grid. Each processing element consists of a processor (usually bit serial), local memory, and communication links to its four neighbors.

For example, the Massively Parallel Processor (MPP) [26] consists of a square grid of 128×128 processors, each having up to 64K bits of memory and interconnectivity to its four nearest neighbors. The processors at one side of the grid can optionally be connected to the other side in different configurations (eg. toruses [33]). The MPP is particularly well-tailored to real-time image analysis where a constantly updated image is presented to the system. Without this continual flow of data, it is doubtful that the parallel system could be kept continuously busy. Theoretically, MPP could offer more flexibility then systolic arrays and can be configured into a pyramid structure.

Proposed Neural Network Architecture

The neural network architecture for computing textural segmentation is presented in Figure 1. The parallel computing architecture proposed consists of three major components: feature extraction network; local boundary detection network; and a higher-order (macro) texture discrimination network. Interactions between these networks results in the segmentation of the textured image.

Feature Extraction Network

In our computing architecture, reflected light intensities from the scene are first sensed and preprocessed by our analog of the vertebrate retina. For our purposes here, the *retina* is reduced to one layer of photoreceptor nodes. These nodes are modeled after vertebrate cones which possess center-surround antagonism in their receptive field organization [34,35]. To reduce the computational complexity of our model, we use the output of these nodes to drive our feature extraction network.

Edge detectors are modeled after *simple* cortical cells [16]. These cells have receptive fields composed of two areas. One type has a receptive field designed to optimally respond to a narrow slit (or bar) of light in the center with dark surrounding flanks or a dark bar in the same place surrounded by light flanks. A second type has a receptive field which optimally responds to an edge with light on the left and darkness on the right (and visa-versa). The edge detectors implemented in our architecture resemble the latter type of simple cell except for the following difference — an edge detector encodes only the orientation of a discontinuity and not the direction of contrast. Edge detectors are constructed in the following manner. For each edge orientation, there are two nodes which are sensitive to the same orientation but to opposite directions of contrast. These nodes are formed by summing the output signals from a sequence of spatially displaced but overlapping photoreceptor nodes (Figure 2). An edge detector receives excitatory input signals from these nodes and encodes the existence of an oriented edge while disregarding the edge's direction of contrast. Edge detectors produce graded responses with a maximum occurring when an edge perfectly aligns with its orientational tuning. Edge detectors of different orientational sensitivities overlooking the same region of the image are grouped together into a column. Columns, covering the entire visual field, form a 3-D cubic structure. These columns are not structurally identical to the vertical slabs reported by Hubel and Wiesel [16] in the visual cortex, although they share a common functionality. This simplification was necessary to reduce the computational complexity of our simulations (for similar approaches see [36,37]).

Edge information can be employed to construct more complex texture elements such as line segments, their terminators, and corners. Although these higher-level features have been previously suggested by Marr [11], Julesz [8], and Walters [37], there is no conclusive neurophysiological explanation of their underlying mechanisms. We are currently investigating solutions to these problems.

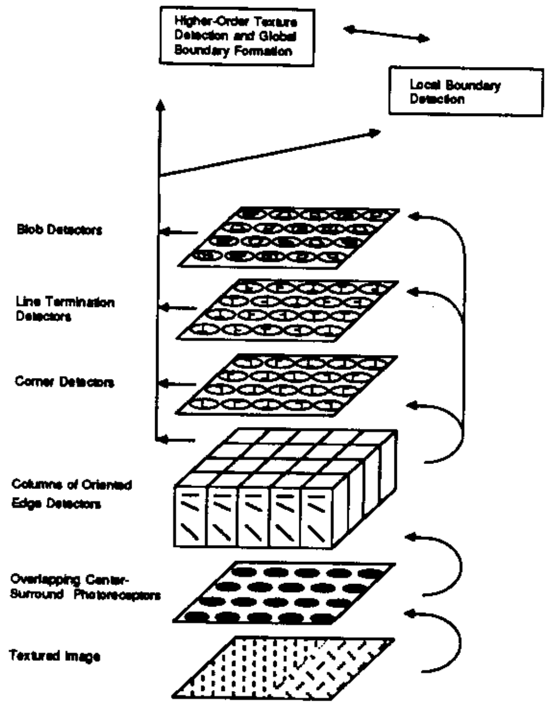

Higher-Order Texture Detection and Global Boundary Formation
Local Boundary Detection
Blob Detectors
Line Termination Detectors
Corner Detectors
Columns of Oriented Edge Detectors
Overlapping Center-Surround Photoreceptors
Textured Image

Figure 1: Neural network architecture for textural segmentation.

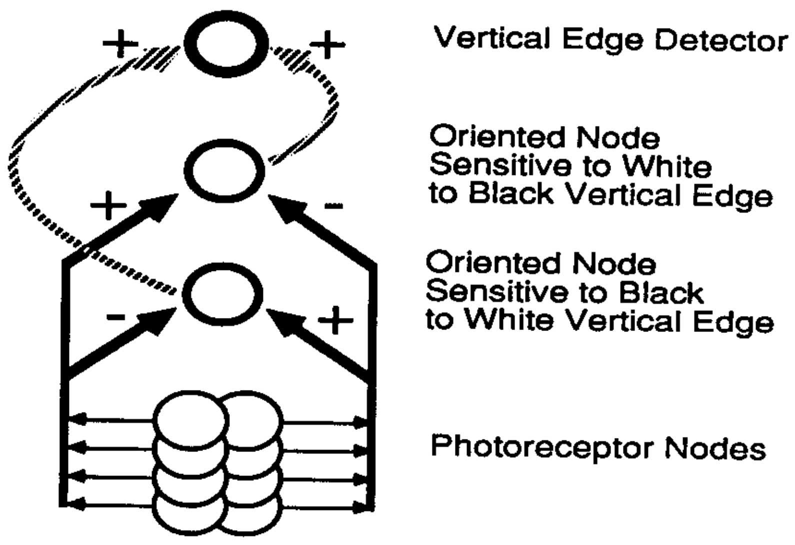

Vertical Edge Detector
Oriented Node Sensitive to White to Black Vertical Edge
Oriented Node Sensitive to Black to White Vertical Edge
Photoreceptor Nodes
+
+
+
-
-
+

Figure 2: Construction of a 90° edge detector.

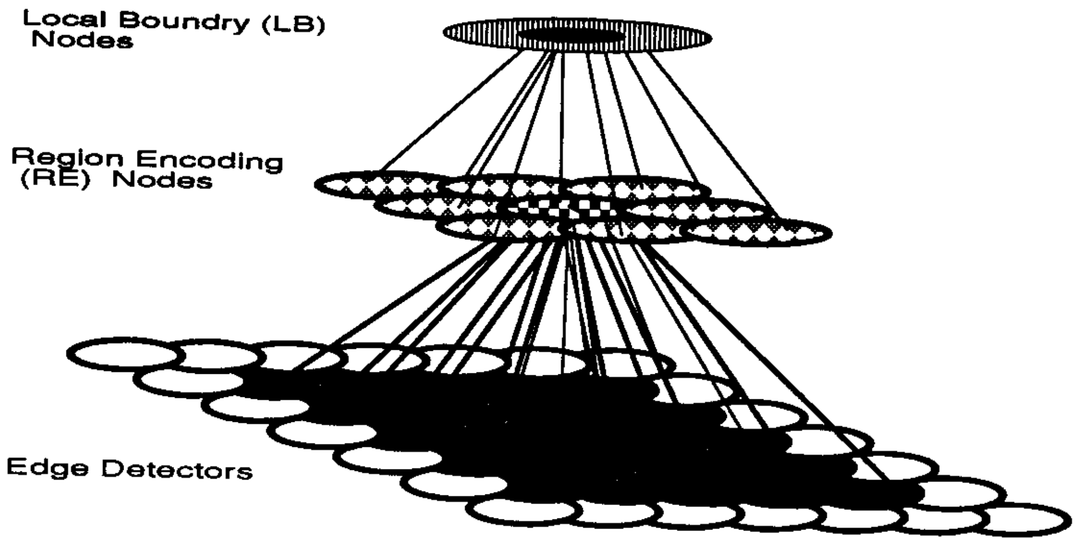


Figure 3: Construction of region encoding (RE) and local boundary (LB) nodes.

Local Boundary Detection Network

The feature extraction network presented provides the remaining layers of the architecture with a preselected set of primitive texels. The next step in the segmentation process is to locate the perimeter of regions that are uniform in texture properties. These local boundaries serve three important functions: 1) delineating uniformly textured regions; 2) linking uniformly textured regions into more complex textural groupings; and 3) linking local boundaries into global boundaries.

Gestalt rules for linking include similarity, proximity, closure, and collinearity. Grouping based on proximity can be overridden by grouping based on intensity or color similarity [38], although it is not clear whether color (hue) similarity prevails over intensity similarity. Grouping based on collinearity is an effective linking rule, however modest variations in intensity or color, inducing rival pairings, can eliminate pairings based on collinearity or orientation. These Gestalt principles have a very natural neural implementation. Grouping mechanisms translate proximity into terms of receptive field size, while "type" similarity is encoded by explicit connectivity to a particular type of node. For example, if the desired similarity measure is orientation, then one possible grouping mechanism would be to provide a physical connection between nodes possessing the same orientational tuning.

Computation of local boundaries requires two steps. First, primitive texels are grouped together using Gestalt principles (similarity, proximity, etc.) [39]. Several textural properties on which to group texels have been proposed [6,40]. These include coarseness, contrast, directionality, line-likeness, regularity, roughness, density, orientation, phase, and frequency. The second step involves the detection of the perimeter of these uniformly textured regions. These perimeters can be located by detecting the discontinuities in the regions of grouped texels. The discontinuities serve as local boundaries encoding the existence of a uniformly textured region. In the discussion below, we will limit ourselves to the textural property of orientation — the predominant feature of response patterns of simple and complex cells [16,19].

Edge detectors of the same orientation are processed by overlapping region encoding (RE) nodes. Collectively, RE nodes isolate areas of similar edge orientation which, by our definition, are primitive texels. A single RE node gathers activity from a region of edge detectors and from a neighborhood of adjacent RE nodes (see Figure 3). A functional analogy can be made between an RE node and the *complex* cells found in the visual cortex [16]. While a simple cell localizes the orientation of a bar of light to a particular position determined by its receptive field, a complex cell signals the abstract concept of orientation without strict reference to position [16]. Hubel and Wiesel's model of a complex cell can be constructed by summing together input from a neighborhood of simple cells. Similarly, an RE node receives input from a neighborhood of edge detectors. However, an RE node also receives input from neighboring RE nodes. This implementation was motivated by the need to propagate orientation information, employed in grouping similarly oriented texels, across the layer of RE nodes.

It is easy to see that an RE node with a receptive field covering an active edge detector will become excited. When there are no active edges within its receptive field, the RE unit can be partially active because of the signals from adjacent RE nodes which exponentially decay over a radial distance. As a consequence of the exponential decay, only the areas around active edge detectors are grouped together to form uniform regions of edge orientation.

Once we have layers of nodes encoding regions with uniform textural properties (in this case edge orientation), the next step is to locate the boundaries formed by these regions. These layers of RE nodes are organized in a columnar structure similar to that of the edge detector nodes (see Figure 1). For each layer of RE nodes there exists a layer of local boundary nodes (LB). LB nodes have center-surround receptive fields and detect discontinuities in the RE layer (see Figure 3). The detection of local boundaries is aided by global analysis of the input image through a second, parallel channel. This second channel processes parafoveal information originating in W-type retinal ganglion cells that compute gross luminance information throughout the image. Nodes modeled after neurons found in visual area 19 coarsely delineate regions of uniform luminance.

Discrimination of higher-order textures

Regions with complex figural texels are segmented by the higher-order texture discrimination network. For example, a herringbone texture can appear as alternating columns of obliquely oriented lines, or as columns of arrows all pointing up or down (see Figure 4a). The herringbone texture can be thought of as possessing a more complex structure (macrostructure) that is composed of primitive texels (microstructure). Nothdurft has shown that there is a strong similarity between macrostructure and microstructure texture discrimination [41]. The higher-order texture discrimination network is based upon the premise that grouping mechanisms employed in the discrimination of simple textures can also be used to discriminate textures of greater structural complexity. Regions composed of different textural properties are linked together to form more complex textural features (macrostructures). Local and global differences in these macrostructures or their organization are enhanced through competitive and cooperative mechanisms and linked together to form boundaries. These boundaries are aggregated to produce the final segmentation of the textured image.

The feature extraction network and the local boundary detection network are both modeled as data-driven, bottom-up processes, while the higher-order texture discrimination network needs to use top-down processing, as demonstrated by the herringbone example above. One task that these mechanisms must be capable of performing is uncovering global regularities. For example, discovering that there are alternating columns of 45° and 135° edges throughout the image. This information can then be used to initiate linking of adjacent columns, resulting in the emergence of columns of arrows all pointing up or down. These networks must also provide grouping mechanisms with feedback signals, indicating whether the elements just grouped resemble known figural elements. More complex Gestalt linking principles, such as good continuation and common fate, are involved in these tasks.

Simulation Results

The connectionist architecture for textural segmentation described above is capable of detecting edge texels and employing them to uncover local boundary contours enclosing uniformly textured regions (based on orientation information only). We have simulated the architecture on a Sun 3/260 workstation using SFINX (Structure and Function In Neural Connections), a neural network simulator developed at UCLA [42]. By reducing the complexity and robustness of the architecture we reduced lengthy simulation times. We used 128×128 grey scale images of natural scenes (textures) as input to the system. The photoreceptor nodes were approximated by a difference of gaussians (DOG) [17] using 1.6 for the ratio of σ_1/σ_2 [43] and a center support diameter of 3 pixels. A set of twelve edge detectors, oriented at every $15°$, was used. For each edge orientation, two nodes sensitive to the same orientation but to opposite directions of contrast were used. These nodes were formed by summing the output signals from a sequence of spatially displaced but overlapping photoreceptor nodes. An edge detector received excitatory input signals from these nodes and encoded the existence of an oriented edge, while disregarding the edge's direction of contrast. The nodes sending excitatory signals to an edge detector (i.e., nodes sensitive to orientation and to the direction of contrast) were modeled as gradient masks (with gaussian smoothing) applied to an 11×11 group of photoreceptor nodes. Nodes responsible for local linking of edge texels of similar orientation into uniform regions received excitatory input from a 5×5 array of overlapping edge detectors, and excitatory support from their neighbors. Local boundary nodes were modeled as center-surround antagonistic receptive fields approximated by a DOG using 1.6 for the ratio of σ_1/σ_2 and a center support diameter of 1 RE node. All simulation results presented in this paper are displayed as grey scale images with *black* representing a zero response and *white* representing a maximal response.

Consider the Brodatz texture presented in Figure 4a [44]. This herringbone texture is composed of alternating columns of obliquely oriented lines (i.e., $45°$ and $135°$ lines). The response of the photoreceptor layer is presented in Figure 4b. Edge detector responses are shown in Figures 4c-d. Region encoding units sensitive to different orientations group edges of similar orientation into uniform regions (see Figure 4e-f). Layers of local boundary nodes then signal the perimeter of the uniformly textured regions (Figures 4g-h).

The simulation results presented above demonstrate the textural segmentation capabilities of the architecture. To better analyze the robustness of our architecture, we performed numerous simulations using textures taken from Brodatz [44]. These simulations made apparent the shortcomings of our region encoding nodes. RE nodes are responsible for grouping textural features into uniformly textured regions (sections of the image which share common featural properties, currently just orientation). The mechanism employed by these units consisted of gathering support from underlying edge detectors ($edge_{neighborhood}$) and support from neighboring RE nodes ($RE_{neighborhood}$). The simulations revealed that the size of neighborhood support $RE_{neighborhood}$ played a *critical* role in grouping the edges into uniform regions (this is discussed in detail in the next section).

While the architecture above detected boundaries which provided a coarse segmentation of microtexture possessed by the image, there is still the need for a mechanism which will continue the segmentation process and uncover the image's macrostructure — columns of arrows all pointing up or down. We are currently investigating the mechanisms necessary for segmenting these higher-order textures.

Conclusions and Future Work

Our attempt to model a neural network architecture for textural segmentation is evolutionary in nature. Hence, further developments depend on inferences derived from experimental results. Careful analysis of

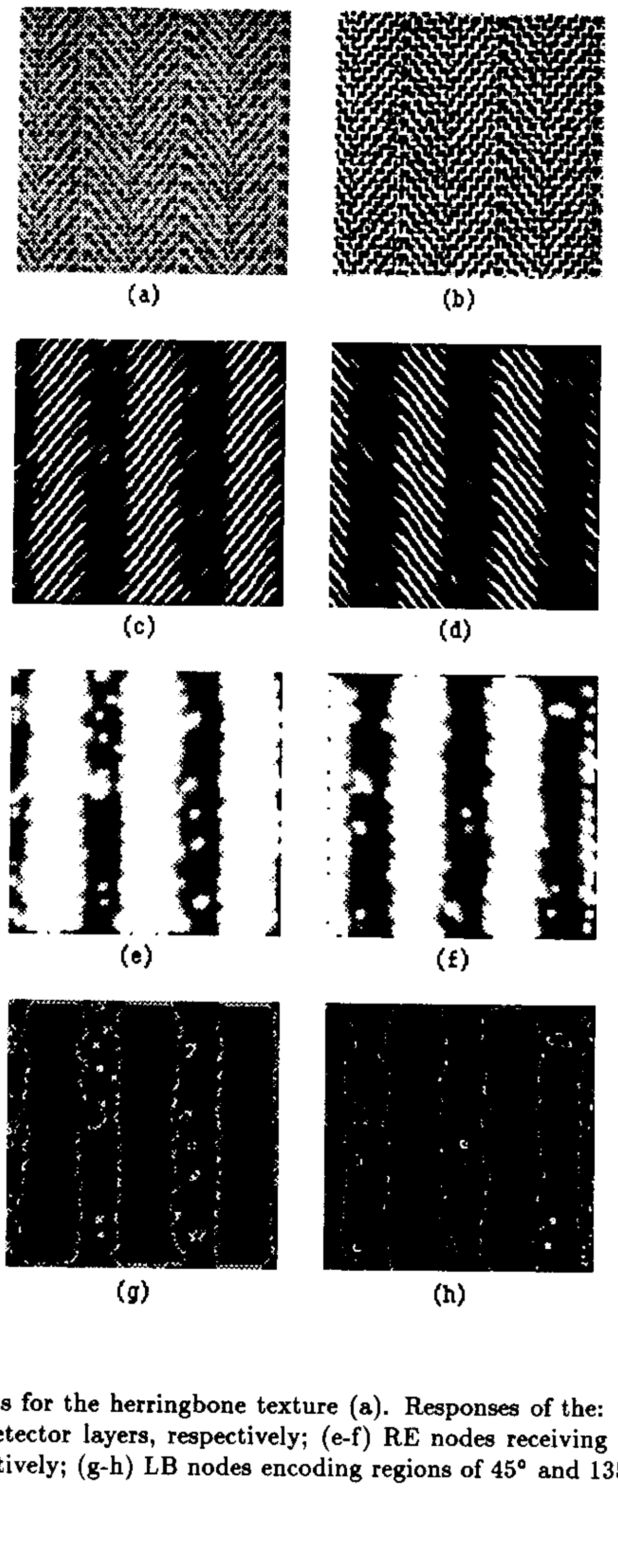


Figure 4: Simulation results for the herringbone texture (a). Responses of the: (b) photoreceptor layer; (c-d) 45° and 135° edge detector layers, respectively; (e-f) RE nodes receiving input from the 45° and 135° edge detectors, respectively; (g-h) LB nodes encoding regions of 45° and 135° edges, respectively.

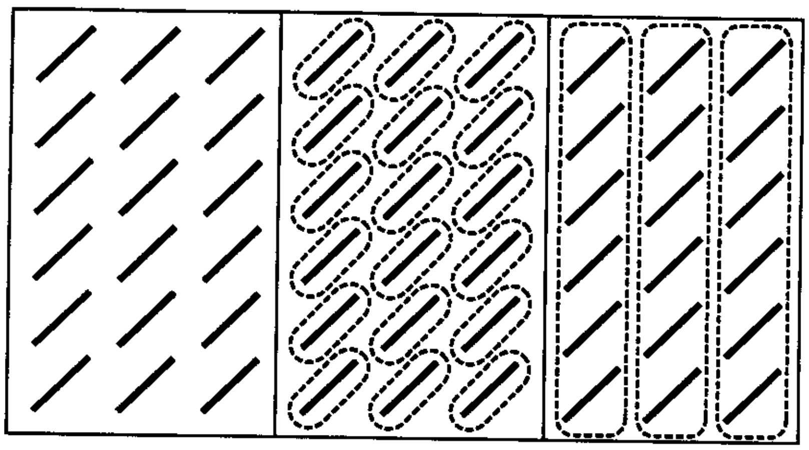


Figure 5: Possible groupings of 60° edge detector responses.

our architecture and the simulation results of numerous textures suggests the following course of research: feature detection, aggregation of multiple features into more complex textural elements, and the development of grouping nodes which can adapt their receptive field sizes and shapes based on the underlying spatial distribution of features.

In the area of feature detection, mechanisms need to be developed to detect line terminations, corners, and blobs. How should these detectors be constructed? One possibility is concatenate the responses of edge detectors spatially arranged to resemble a corner. Another possibility is to hardwire the corner detector directly to the photoreceptor inputs.

Multiple feature aggregation also poses challenging problems. What features should be combined? For example, should edge information be combined with corner or blob information, and which one should take precedence. How can features be combined over different spatial scales? Another problem related to aggregation is the embedding Gestalt grouping principles into neural structures. Most of our past effort was focused on use of proximity between features. In the continuation of our work, we will investigate combinations of Gestalt grouping principles.

Our current effort is on the development of robust grouping mechanisms. Consider the response of 60° edge detectors presented in Figure 5a. How can these responses be grouped? One possible grouping is to form thin elongated blobs, representing the individual stripes (see Figure 5b). Another possibility is to group these responses into a collection of stripes (a vertical column of stripes, see Figure 5c). In the former case, the edges are separated by a distance Δ_1 (i.e., the thickness of the stripe). In the latter case, the collection of edge responses (a stripe) are separated by distance Δ_2 (inter-stripe distance).

This example illustrates the inherent relationship between the size of a grouping node's receptive field and the spatial distribution of features (in this case edge responses). When the node's receptive field is much larger than the feature, its response is very small. When the node's receptive field is much smaller than the feature, its response is saturated. Hence, the ideal situation occurs when the receptive field matches the size of the feature. How can we detect this situation? One possible solution is to monitor the derivative of the grouping node's response as its receptive field is adaptively changing.

The receptive field size of a node designed to detect individual stripes would be much smaller than the receptive field of a node designed to detect the column of stripes. In either case, the receptive field dimensions are influenced by the size of the feature to be detected and the distribution of the feature. How can these groupings be performed? One approach is to develop a family of grouping nodes which operate on a range of feature sizes and distributions. For example, elongated Gaussian receptive fields of different spatial scales and orientations can be used to group textural features. While this spatial frequency channel-like scheme [14,45] can be implemented in a hierarchical computing structure, it poses challenging problems. What are the scales? How many orientations? How do we integrate the information across the grouping scales?

Our proposed approach is to develop grouping nodes which have adaptable receptive fields which adjust their shape and size based on the underlying features. What kind of mechanisms are involved in modifying a node's receptive field? Adaptive mechanisms based only on local information result in receptive fields which can grow to enclose a feature cluster (see Figure 5b). For example, one scheme to achieve this grouping is to have each grouping node grow its receptive field as long as its response is increasing.

However, grouping mechanisms based on local information encounter difficulties when the desired grouping is a collection of feature clusters (see Figure 5c). This problem can be alleviated somewhat, if some *global* information about the feature space is available. This global information is not derived through the analysis of the entire feature space, but rather, from nodes possessing larger receptive fields than individual grouping nodes. Since these nodes with larger receptive fields have a more global view of the feature space, they provide valuable information about the feature space surrounding individual the grouping nodes. This feedback information can be used by the grouping node to adjust its receptive field size and shape to adapt to the underlying feature space. Julesz has also suggested the use of global information to adjust the spatial scale of grouping mechanisms [46]. His revised texton theory calls for global operations performed by the preattentive system to determine the proper spatial units for grouping (i.e., the size of grouping neighborhoods).

Acknowledgements

We sincerely acknowledge support from MICRO-Hughes Malibu Research Laboratory grant #541122-57442, ONR grant #N00014-86-K-0395 and ARO grant #444251-25519. Support for the UCLA Machine Perception Laboratory environment is provided in part by generous equipment grants from IBM and Hewlett Packard.

References

[1] M. D. Levine. *Vision in Man and Machine.* McGraw-Hill, New York, 1985.

[2] J. K. Hawkins. Textural properties for pattern recognition. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 347–370, Academic Press, New York, 1970.

[3] P. Burt. The pyramid as a structure for efficient computation. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 6–35, Springer Verlag, New York, 1984.

[4] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[5] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(6):610–621, 1973.

[6] K. Laws. *Textured image segmentation.* PhD thesis, University of Southern California, Los Angeles, 1980. USCIPI Report No. 940.

[7] L. S. Davis, S. R. Johns, and J. K. Aggarawal. Texture analysis using generalized co-occurrence matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):251–258, 1979.

[8] B. Julesz. A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological Cybernetics*, 41:131–138, 1981.

[9] A. Gagalowitz. A new method for texture field synthesis: some applications to the study of human vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):520–533, 1981.

[10] F. M. Vilnrotter, R. Nevatia, and K. E. Price. Structural analysis of natural textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):76–89, 1986.

[11] D. Marr. *Analyzing Natural Images, A Computational Theory of Texture Vision.* Technical Report AI Memo 334, MIT, 1975.

[12] J. Beck, K. Prazdny, and A. Rosenfeld. A theory of textural segmentation. In J. Beck and B. Hope, editors, *Human and Machine Vision*, pages 1–38, Academic Press, New York, 1983.

[13] D. Wermser and C. Liedtke. Texture analysis using a model of the visual system. In *Proceedings of 6th International Conference on Pattern Recognition*, pages 1070–1080, Munich, 1982.

[14] J. Beck, A. Sutter, and R. Ivry. Spatial frequency channels and perceptual grouping in texture segregation. *Computer Vision, Graphics, and Image Processing*, 37:299–325, 1987.

[15] B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, IT-8:84–92, 1962.

[16] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

[17] R. W. Rodieck. *The Vertebrate Retina, Principles of Structure and Function.* Freeman, San Francisco, 1973.

[18] D. Reddy and R. Hon. Computer architecture for vision. In G. Dodd and L. Rossol, editors, *Computer Vision and Sensor-Based Robots*, pages 169–184, Plenum Press, New York, 1979.

[19] H. C. Nothdurft and C. Y. Li. Texture discrimination: representation of orientation and luminance differences in cells of the cat striate cortex. *Vision Research*, 25(1):99–113, 1985.

[20] A. P. Reeves. Parallel computer architectures for image processing. *Computer Vision, Graphics, and Image Processing*, 25(1):68–88, January 1984.

[21] A. Rosenfeld. Pyramid machines: some advantages. In *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 159–161, Pasadena, Calif., October 1983.

[22] H. T. Kung and S. W. Song. A systolic 2-D convolution chip. In *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 159–160, Hot Springs, November 1981.

[23] H. Kung, M. Ruane, and D. W. L. Yen. Two-level pipelined systolic array for multidimensional convolution. *Image and Vision Computing*, 1(1):30–36, February 1983.

[24] W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, 1985.

[25] *iPSC User's Guide.* Intel Corporation, No. 175455-004, 1986.

[26] K. Batcher. Design of a massively parallel processor. *IEEE Transactions on Computers*, C-29(9):836–840, September 1980.

[27] A. Rosenfeld. Some useful properties of pyramids. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 2–5, Springer-Verlag, New York, 1984.

[28] N. Ahuja and S. Swamy. Multiprocessor pyramid architectures for bottom-up image analysis. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 38–59, Springer-Verlag, New York, 1984.

[29] L. Uhr. A 2-layered SIMD/MIMD parallel pyramidal array/net. In *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 209–216, Hot Springs, November 1981.

[30] C. R. Dyer. A VLSI pyramid machine for hierarchical parallel image processing. In *Proceedings of IEEE Conference on Pattern Recognition and Image Processing*, pages 381–386, Dallas, Texas, 1981.

[31] H. Kung. On the implementation and use of systolic array processors. In *IEEE International Conference on Computer Design*, pages 370–373, Port Chester, November 1983.

[32] J. J. Little. *Parallel Algorithms for Computer Vision.* Technical Report AI Memo 928, MIT, 1986.

[33] J. Potter. Continuous image processing on the MPP. In *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 51–56, Hot Springs, November 1981.

[34] D. A. Baylor, M. G. Fuortes, and P. O'Bryan. Receptive fields of cones in the retina of the turtle. *J. Physiol.*, 214:265–294, 1971.

[35] J. Skrzypek. A computing structure for a spatial-context model of lightness constancy. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, November 1982. Tucson, Arizona.

[36] S. Zucker. Cooperative grouping and early orientation selection. In O. Braddick and A. Sleigh, editors, *Physical and Biological Processing of Images*, Springer Verlag, New York, 1983.

[37] D. Walters. Selection and use of image features for segmentation of boundary images. In *Proceedings of CVPR*, 1986.

[38] K. A. Stevens and A. Brookes. Detecting structure by symbolic constructions of tokens. *Computer Vision, Graphics, and Image Processing*, 37:238–260, 1987.

[39] M. Wertheimer. Untersuchungen zur lehre von der gestalt. *Psychologische Forschung*, 4:301–350, 1923. Translation in A Source Book of Gestalt Psychology, W. D. Ellis, ed., New York: Harcourt, Brace, 1938.

[40] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(8):460–473, 1978.

[41] H. C. Nothdurft. Discrimination of higher-order textures. *Perception*, 14:539–543, 1985.

[42] E. Paik, D. Gungner, and J. Skrzypek. UCLA SFINX - A neural network simulation environment. In *Proceedings of the IEEE First Annual International Conference on Neural Networks*, 1987. San Diego, California, June 21-24.

[43] D. Marr and E. Hildreth. Theory of edge detection. *Proc. R. Soc. London B*, 207:187–217, 1980.

[44] P. Brodatz. *Textures, A Photographic Album for Artists and Designers.* Dover Publications Inc., New York, 1966.

[45] A. C. Bovik, M. Clark, and W. S. Geisler. Computational texture analysis using localized spatial filtering. In *Proceedings of IEEE Computer Society Workshop on Computer Vision*, pages 201–206, Miami Beach, Florida, 1987.

[46] B. Julesz. Texton gradients: the texton theory revisited. *Biological Cybernetics*, 54:245–251, 1986.