ELSEVIER

Neurocomputing 14 (1997) 241–272

NEUROCOMPUTING

Multiscale image segmentation using a hierarchical self-organizing map

Suchendra M. Bhandarkar [a,*], Jean Koh [b], Minsoo Suk [b]

[a] *Department of Computer Science, University of Georgia, Athens, GA 30602-7404, USA*
[b] *Department of Electrical and Computer Engineering, Syracuse University, Syracuse, NY 13244-1240, USA*

Received 24 April 1995; accepted 31 March 1996

Abstract

Multiscale structures and algorithms that unify the treatment of local and global scene information are of particular importance in image segmentation. Vector quantization, owing to its versatility, has proved to be an effective means of image segmentation. Although vector quantization can be achieved using self-organizing maps with competitive learning, self-organizing maps in their original single-layer structure, are inadequate for image segmentation. A hierarchical self-organizing neural network for image segmentation is presented. The Hierarchical Self-Organizing Map (HSOM) is an extension of the conventional (single-layer) Self-Organizing Map (SOM). The problem of image segmentation is formulated as one of vector quantization and mapped onto the HSOM. By combining the concepts of self-organization and topographic mapping with those of multiscale image segmentation the HSOM alleviates the shortcomings of the conventional SOM in the context of image segmentation.

Keywords: Image segmentation; Self-organizing map; Neural networks; Computer vision

1. Introduction

Image segmentation is a process of partitioning an input image into disjoint subregions which individually satisfy the properties of homogeneity and connectivity [2]. A region is considered homogeneous if all its pixels are homogeneous with respect to one or more pixel attributes such as intensity, texture, color, range, etc. A region is considered connected if there exists a connected path between any two pixels within the region. The segmented image, which is a result of grouping the elements (typically

* Corresponding author. Email: suchi@cs.uga.edu.

PII S0925-2312(96)00048-3

pixels) of an input image into semantically meaningful entities, is generally considered to be the highest domain-independent or data-driven abstraction of the input image. A segmented image is the primary input to high-level vision which subsequently utilizes domain-specific knowledge to interpret and analyze the image contents.

It is widely acknowledged by neurophysiologists, perceptual psychologists and computer vision scientists that vision could be looked upon as a set of information processing tasks that detect and extract information from the visual field, construct and integrate hypotheses, and build scene descriptions at successively higher levels of abstraction [20]. Visual structures and algorithms that are capable of representing and processing visual data at multiple scales of abstraction have a strong appeal from a biological point of view. The complexity analysis of most biological (in particular, human) visual systems strongly indicates that hierarchical or multiscale internal representations and processing techniques are employed during scene interpretation. From the computer vision scientist's point of view, hierarchical or multiscale representations and processing techniques offer a means of integrating local and global scene information and also alleviating the combinatorial complexity of scene interpretation [13].

Furthermore, in most computer vision tasks, the appropriate choice of scale is a crucial issue. In most images, the objects appear with varying sizes, at varying positions and orientations and at varying degrees of spatial resolution. Consequently, there often does not exist a single scale of spatial resolution that could be deemed appropriate for analysis of the entire image. A choice of a single scale may prove too fine in certain portions of the image resulting in unnecessary detail, noise and clutter and too coarse in other portions of the image resulting in loss of desired detail. Clearly, a multiscale analysis of the image is necessary which naturally entails a segmentation technique that is capable of generating a multiscale representation of the image data.

1.1. Image segmentation

As previously mentioned, the goal of image segmentation is to partition an image into mutually exclusive and exhaustive regions such that each region is spatially contiguous and the pixels within the region are homogeneous with respect to a predefined criterion. Some widely used homogeneity criteria include values of intensity, texture, color, range, surface normal and surface curvature(s).

Consider an input image of the form $\{f(x,y);\ x=1,2,\ldots,N_x,\ y=1,2,\ldots,N_y\}$. When $f(x,y)$ is a scalar function, its value reflects the measurement of a single scene attribute of interest, typically intensity, texture or range. When $f(x,y)$ is a vector function, it reflects the measurement of several scene attributes. For example, the components of $f(x,y)$ may represent values of intensity, texture, range, surface normal or surface curvature at pixel (x,y). Let $R_1,R_2,\ldots,R_M$ be the regions resulting from the segmentation of an image where each R_i, $1 \le i \le M$ is a connected region. Also, let the predicate P represent the homogeneity criterion such that $P(R_i)=T$ for all i, but $P(R_i \cap R_j)=F$ for any $i \neq j$. For an n-dimensional vector $f(x,y)$, we define an $(n+2)$-dimensional vector for each location in the image, $\mathbf{x}_i=(x,y,f(x,y))$, $i=1,2,\ldots,N_x \times N_y$. We can define the regions in the segmented image in terms of these vectors as $P'(R_i)=T$ and $P'(R_i \cap R_j)=F$ for $i \neq j$, where P' is a new predicate that

incorporates both connectivity and homogeneity defined over a subspace of the $(n+2)$-dimensional space.

Pal and Pal [23], Haralick and Shapiro [9], Fu and Mui [8], and Sahoo et al. [26] present excellent surveys of existing image segmentation techniques. Image segmentation techniques can be broadly classified as based on: (1) edge detection; (2) region or surface growing; (3) gray level thresholding; (4) iterative pixel classification using relaxation, Markov Random Fields (MRFs), and Gibbs Random Fields (GRFs); and (5) feature vector clustering or vector quantization. For each of the aforementioned segmentation techniques, statistical, fuzzy, syntactic, neural network-based or knowledge-based approaches are conceivable.

1.2. Vector quantization

Owing to its versatility, vector quantization has proved to be a very effective model for image segmentation [1]. Essentially, vector quantization is a process of partitioning an n-dimensional vector space into M regions so as to optimize a criterion function when all the points in each region are approximated (or represented) by the representation vector $\mathbf{X}_i$ associated with that region. The problem, therefore, is to find an appropriate partition and an appropriate set of representation vectors that would result in an optimum of the criterion function. The optimization could be posed as the minimization of a criterion function that measures the overall distortion arising from the resulting quantization or as the maximization of a criterion function based on entropy which ensures that all representation vectors are used with equal frequency. Determining such an optimal quantizer entails a training procedure which essentially learns the probability distribution of the input data. A typical learning procedure involves two steps for each input vector $\mathbf{v}$:

1. identify the best representation vector $\mathbf{u}$ from among the current set of representation vectors, and
2. update the representation vector $\mathbf{u}$ so as to incorporate the current input vector $\mathbf{v}$.

Consider the problem of image segmentation where one wants to divide an input image into M subregions, $R_1, R_2, \ldots, R_M$ for a prespecified value of M. This problem could be formulated as a vector quantization problem:

1. Initialization: Select M initial representation vectors, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_M$, corresponding to the M regions.
2. Select a point, $\mathbf{x} = (x, y, f(x,y))$ and find the representation vector $\mathbf{X}_k$ from among $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_M$, which is closest to $\mathbf{x}$, i.e. $d(\mathbf{x}, \mathbf{X}_k)$ is minimum.
3. Update the representation vector $\mathbf{X}_k$.
4. Repeat Steps 2 and 3 for all pixels in the image.

The updating rate is a constant of proportionality which is used to weigh the amount by which the representation vector is updated towards the input vector. To enforce convergence, the updating rate in Step 3 tends to 0 with each successive iteration. The

function $d(\cdot,\cdot)$ is an appropriately defined distortion measure which is chosen to reflect the homogeneity criterion as well as the connectivity requirement. The vector quantization procedure does not strictly guarantee the connectivity of each region, but if $d(\cdot,\cdot)$ is chosen judiciously, most of the regions in the segmented image would be connected.

1.3. Vector quantization and the competitive learning neural network

There are two processes involved in vector quantization: one is the training process which determines the set of codebook vectors according to the probability distribution of the input data; the other is the encoding process which assigns input vectors to the codebook vectors (i.e. representation vectors). Vector quantization has a natural formulation and implementation in terms of the Competitive Learning Neural Network (CLNN) [21]. The CLNN structure for the encoding process is fairly straightforward. Consider an n-dimensional input vector space and a distortion measure $d(\mathbf{x},\mathbf{y})$ defined in this space. Let the size of codebook be M and the codebook vectors be $\mathbf{w}_i$, where $i = 1,\ldots,M$. Consider a neural network with M neural units and assign the i-th codebook vector to the weight vector associated with neural unit i. In order to encode any given vector $\mathbf{x}$, $\mathbf{x}$ is fed in parallel to all the M neural units. Each of these units computes the distance d_i (distortion) between the input vector and its weight vector given by $d_i = d(\mathbf{x},\mathbf{w}_i)$. The input vector is then encoded as the index j such that $d_j = \min_i d_i$. All the above computations can be carried out in parallel. However, a more fundamental benefit of mapping the process of vector quantization onto a CLNN is that many neural network learning algorithms that have been developed can be readily adapted to the training process. An adaptive version of the Linde–Buzo–Gray (LBG) algorithm [18] for the training of vector quantizers can be implemented using competitive learning.

Let us assume that the M neural units in the CLNN are initialized with randomly generated weight vectors. Starting with these initial weight vectors, the learning algorithm iterates a certain number of times through the training data, adjusting the weight vectors of the neural units on presentation of each training vector. The algorithm used to adjust the weight vectors is based on competitive learning. The input vector $\mathbf{x}$ is presented to all of the neural units and each unit computes the distance (distortion) between its weight vector and the input vector. The neural unit with the smallest distance is designated as the winning unit and its weight vector is adjusted towards the input vector. As explained above, one can consider weight vectors in the competitive learning neural networks to be the codebook vectors in the vector quantization process and the distance measure to be the distortion measure. Also the adaptation process used in competitive learning can be considered to be analogous to the training process used in vector quantization. The clustering process that follows learning can be considered to be analogous to the encoding process in vector quantization.

1.4. Image segmentation and the SOM

The Self-Organizing Map (SOM) [15], which is a member of the larger class of CLNNs, is a popular choice when implementing vector quantization using neural networks. Many examples of the use of the SOM for vector quantization can be found in

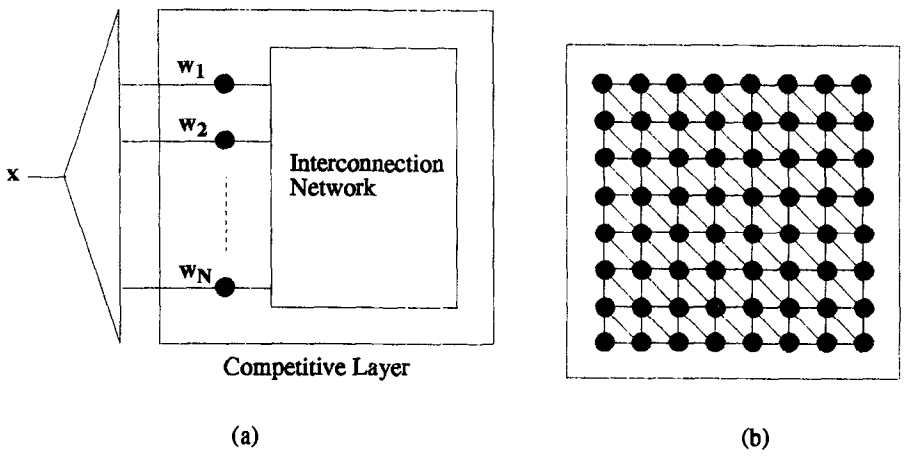


Fig. 1. The network structure of the SOM.

the literature [4,6,12,15,17,22,28]. A typical SOM structure is shown in Fig. 1. It consists of two layers – a layer of input nodes and a competitive layer consisting of neural units called Kohonen's units [15]. It should be noted that the input layer is usually not counted when specifying the number of layers, hence the conventional SOM is referred to as a single-layer SOM. A weight vector is associated with each connection from the input layer to a neural unit. The neural units in the competitive (and cooperative) layer are organized in a regular geometric structure such as a two-dimensional mesh. The neural units are interconnected with their local neighbors with either excitatory or inhibitory connections.

The competitive phase of the learning algorithm employed in the SOM determines a winning neural unit whereas the cooperative phase of the learning algorithm updates the weights of the winning neural unit and the neural units in its neighborhood. If $\mathbf{x}$ is an input vector to the SOM then the learning algorithm for the SOM could be described as follows:

1. The distances between the input vector $\mathbf{x}$ and all the reference vectors (i.e. the weight vectors) are computed using a prespecified distance measure.
2. A winner, i.e. a neural unit whose corresponding weight vector is at a minimum distance from the input vector, is determined.
3. The weight vectors corresponding to the winner and the neural units in its topological neighborhood are updated so as to align them towards the input vector.

Steps 1 and 2 comprise the competitive phase of the learning algorithm whereas step 3 comprises the cooperative phase. The competitive phase is used for the encoding process in vector quantization. Steps 1 through 3 of the learning algorithm, which are used in the training process in vector quantization, are carried out for each input vector that is presented to the SOM. The learning rate is defined to be the constant of proportionality which indicates the extent to which the weight vectors are adjusted

towards a given input vector. The learning rate (updating rate) and the neighborhood size are reduced over the course of the learning process.

The most important property of the SOM is that it represents a *topology-preserving* mapping or a *topographic* mapping. This is a direct consequence of the fact that the weights associated with a winning neural unit *and* those in its neighborhood are simultaneously updated during the learning process unlike other CLNNs. The location of the winning neural unit in the 2-D mesh of neural units comprising the competitive layer of the SOM, conveys some information about the input vector. Winning units which are proximate in the competitive layer correspond to input vectors that are proximate in the input vector space. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are two input vectors with corresponding winning units at locations $\mathbf{r}_1$ and $\mathbf{r}_2$ in the competitive layer then $\mathbf{r}_1$ and $\mathbf{r}_2$ get closer and eventually coincide as $\mathbf{x}_1$ and $\mathbf{x}_2$ are made more and more similar. Conversely, if $\mathbf{r}_1 = \mathbf{r}_2$, one could conclude that the corresponding input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are similar or proximate when viewed in the input vector space. A topographic mapping thus preserves topological relations in the input space while simultaneously performing a dimensionality reduction (i.e. projection) of the input space onto the 2-D mesh of neural units in the competitive layer. A topographic map entails a suitably defined topological neighborhood around a winning neural unit in the competitive layer and also a distance metric associated with the topological neighborhood.

The popularity of the SOM for vector quantization is primarily due to the similarity between the competitive learning process employed in the SOM and the vector quantization procedure. The competitive learning process in the SOM results in weight vectors which correspond to distinct clusters of the input vectors. In fact, the weight vectors can be considered to be the cluster centers of the probability density function of the input data [15]. Representing or approximating an input vector by the weight vector associated with the winning unit in the SOM is equivalent to deriving the minimum-residual-error-approximation to the input vector which is exactly the same result sought by the vector quantization process [1,18].

1.5. Choice of feature vectors for vector quantization

In image segmentation by vector quantization, the homogeneity of a region in the segmented image is enforced by the appropriate choice of feature vectors. In the case of intensity images of the form $I(x,y) = f(x,y)$, the input image is presented to the SOM as a three-dimensional vector $\mathbf{x}_i = (x, y, f(x,y))$. Given the fact that range images are an explicit representation of the scene surface geometry (albeit in sampled form), one expects to use a homogeneity criterion that incorporates the intrinsic geometrical properties of the 3-D surfaces when segmenting range images.

The boundary of a homogeneous region in a range image can comprise of one or more of the following three types of edges: jump edges, crease edges or curvature edges. Jump edges occur where depth values are discontinuous in a range image. It is therefore essential to incorporate depth information (i.e. the range value) as one of the components (i.e. feature attributes) of the feature vector in order to use the position of jump edges as decision boundaries in the clustering or vector quantization process. Crease edges correspond to surface pixels where there is a discontinuity in the surface normal

but continuity in the depth. Curvature edges are characterized by discontinuity of the surface curvature but continuity of the surface normal. Thus, it is necessary to incorporate the unit surface normal and the invariant surface curvature values as feature attributes or components of the feature vector in order to use crease edges and curvature edges, respectively, as decision boundaries between clusters or surface regions. Among the invariant surface curvatures, the mean and Gaussian curvatures have been widely used in range image segmentation algorithms [3]. The position coordinates x and y are also included in the feature vector in order to preserve the spatial connectivity of the subregions in the final segmented image. The input feature vector in the case of a range image is therefore an eight-dimensional vector: $\mathbf{x} = (x, y, z;\ n_x, n_y, n_z;\ H;\ K)$, where (n_x, n_y, n_z) is the unit normal vector and H and K are the mean and Gaussian curvatures, respectively at the point (x, y, z) on the three-dimensional surface.

1.6. Shortcomings of the SOM

One of the shortcomings of the conventional SOM is that the number of neural units in the competitive layer needs to be approximately equal to the number of regions desired in the segmented image. However, it is usually not possible to determine a priori the correct number of regions M in the segmented image. This severely limits the usefulness of the conventional single layer SOM for image segmentation. When the SOM has far fewer neural units than the number of visually distinguishable regions in the input image, the result is an undersegmented image in which several visually distinguishable regions are incorrectly merged into a single region. On the other hand, when the SOM has a much greater number of neural units than the number of visually distinguishable regions in the input image, the result is an oversegmented image in which homogeneous regions are incorrectly split into several regions. In short, the single-layer SOM does not give us direct control over the number of regions in the segmented image. This is a significant shortcoming of the single-layer SOM since the number of regions in the final segmented image is very much dependent on the input scene content, and is difficult to determine a priori. This clearly calls for a neural network that can produce a multiscale segmentation of the input image.

In this paper we a propose a Hierarchical Self-Organizing Map (HSOM) that directly addresses the aforementioned shortcomings of the conventional single-layer SOM. We describe the structure of the HSOM and the learning procedure that would make it suitable for image segmentation. The HSOM is shown to combine the ideas of *self-organization* and *topographic mapping* with those of *multiscale image segmentation*, thus making it distinctly unique from previous approaches to image segmentation using neural networks. The performance of the HSOM is demonstrated with experimental results using both, intensity and range images.

2. Multiscale image segmentation using the HSOM

As previously mentioned, images of real-world scenes contain objects of varying sizes and objects viewed at different distances and from varying viewpoints resulting in

image features at many different scales. Choosing the correct scale for analyzing the image features is critical for image segmentation and subsequent scene analysis. In many instances, there is no a priori basis for selecting a particular scale for feature representation. Also, using a single scale for an entire image may not be appropriate since the scale at which to represent or analyze a particular feature may depend strongly on the presence of other features of different sizes in its spatial proximity. Multiscale image representations are a powerful tool for analyzing image features at multiple scales. An image is decomposed into a set of descriptions each making explicit the image features at a specific scale. Representations of this type are broadly termed as pyramids, multi-resolution images, multiscale images or scale-space images. A considerable amount of research effort has been devoted towards the construction and analysis of multiscale image representations [7,19,29]. We will show that the HSOM achieves multiscale segmentation of an input image. In fact, it will be shown that the HSOM generalizes the conventional notion of *scale* that is defined with respect to the *image* space to one that is defined with respect to the *feature* space.

The HSOM conforms to the idea of regarding the image segmentation process as one of data abstraction where the segmented image is the final domain-independent abstraction of the input image. Consider a hierarchical approach to segmentation: the first stage segments an input image into M_1 regions. These regions could be treated as the first-level abstraction of the input image. The segmentation process is repeated resulting in $M_2, M_3, \ldots$ regions until one reaches the highest level of abstraction which is the trivial case consisting of a single region covering the entire input image. The hierarchical segmentation procedure described above results in a hierarchical structure which we call an *abstraction tree*. The abstraction tree containing all levels of abstraction is depicted in Fig. 2(a). Note that each node in the abstraction tree represents a subregion at a specified level of abstraction.

One could imagine several different ways in which the representation of an input image in the form of an abstraction tree could be used by the high-level vision processes. In fact, one may regard the abstraction tree as the final outcome of low-level vision and use the abstraction tree in its entirety as input to the high-level vision processes. A segmented image could be generated on demand by traversing the abstraction tree in a breadth-first manner starting from the root node until some stopping criterion is met. In the breadth-first traversal, a node in the abstraction tree is expanded if the variance of the feature attribute values of the pixels in the region represented by the corresponding node is larger than a prespecified threshold value. Otherwise, the node is labeled as a closed node and none of its descendants are visited. Regions corresponding to the *closed* nodes constitute a segmented image. Note, that the resulting segmented image usually contains regions from different abstraction levels.

The abstraction tree bears some resemblance to the more familiar quadtree data structure [27] used in several image processing and image analysis algorithms. The root node in the quadtree represents the entire image. The image is divided into four quadrants which are represented by the four child nodes of the root node. The partitioning of a quadrant (i.e. a node) into four subquadrants (i.e. four child nodes) is repeated until the leaf nodes (which represent the individual pixels of the input image) are reached, resulting in a pyramidal data structure. The quadtree is used extensively in

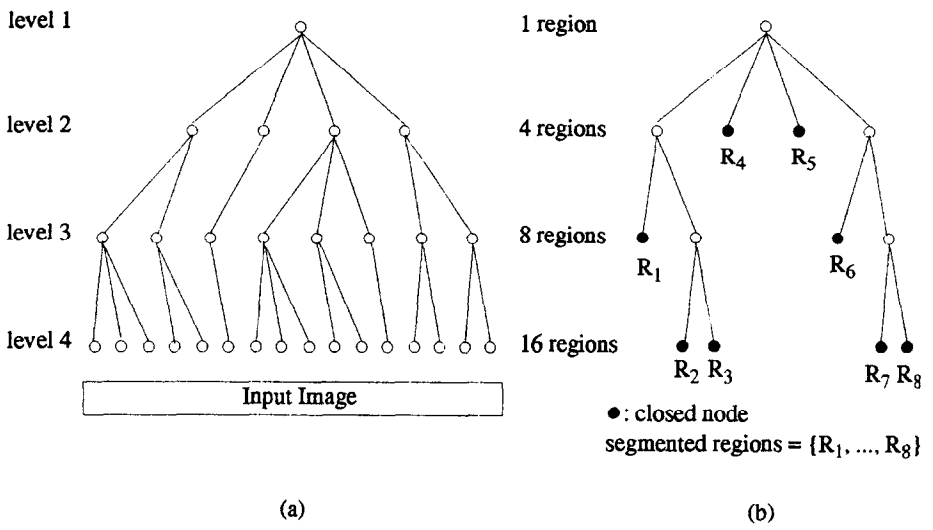


Fig. 2. Abstraction tree. (a) A 3-level abstraction tree, (b) A tree to corresponding the segmented image.

image segmentation algorithms based on the *split-and-merge* technique [24]. However, it should be noted that the abstraction tree is more general than the quadtree. A parent node in the quadtree is restricted to having strictly four child nodes. This restriction causes regions in the segmented image resulting from a split-and-merge algorithm to have a *box-like* appearance. Since this restriction does not apply to the abstraction tree, one does not encounter the occurrence of box-like regions in the segmented image when the abstraction tree is used.

In the following sections we describe how an abstraction tree could be generated from an input image using the HSOM. Note that image segmentation using the conventional single-layer SOM would produce an abstraction of the input image at a prespecified (and somewhat arbitrarily chosen) level of abstraction. In contrast, the segmented image produced by the HSOM would consist of regions belonging to different levels of abstraction where the level of abstraction for a given region would depend on the local scene content. The ability to adapt the level of abstraction to the local scene content without having to make a single (and somewhat arbitrary) choice of a level of abstraction for the entire image is highly desirable. This is the primary advantage of using the HSOM for image segmentation.

In the final segmented image generated by the HSOM, portions of the image where the scene attributes are largely homogeneous are mapped onto neural units in the HSOM at higher levels of abstraction, whereas portions of the image where the scene attributes vary greatly and rapidly (i.e. busy regions in the image) are mapped onto neural units in the HSOM at lower levels of abstraction. Since the HSOM, unlike the SOM, is capable of adapting the level of abstraction in accordance with the local scene content it represents a significant improvement over the conventional single-layer SOM in the context of image segmentation.

2.1. The HSOM structure

The HSOM is organized as a pyramidal structure consisting of multiple layers where each layer resembles the single-layer SOM. Input data arrives at the lowest layer and information flows to the higher layers in a strictly feed-forward manner. An intermediate buffer unit converts the output from any given layer to the input for the succeeding layer. A hierarchical structure is constructed by stacking the layers in a pyramidal fashion where the number of neural units decreases gradually as information propagates to the higher layers in the HSOM. Since, the number of representation vectors to be formed in a given layer is typically proportional to the number of neural units in that layer, in a higher layer of the HSOM which contains fewer neural units, each weight vector represents a larger cluster in input vector space, i.e. a higher level of abstraction of the input image.

The HSOM consists of multiple layers each of which implements competitive learning. The input layer receives input from the external world and propagates the input

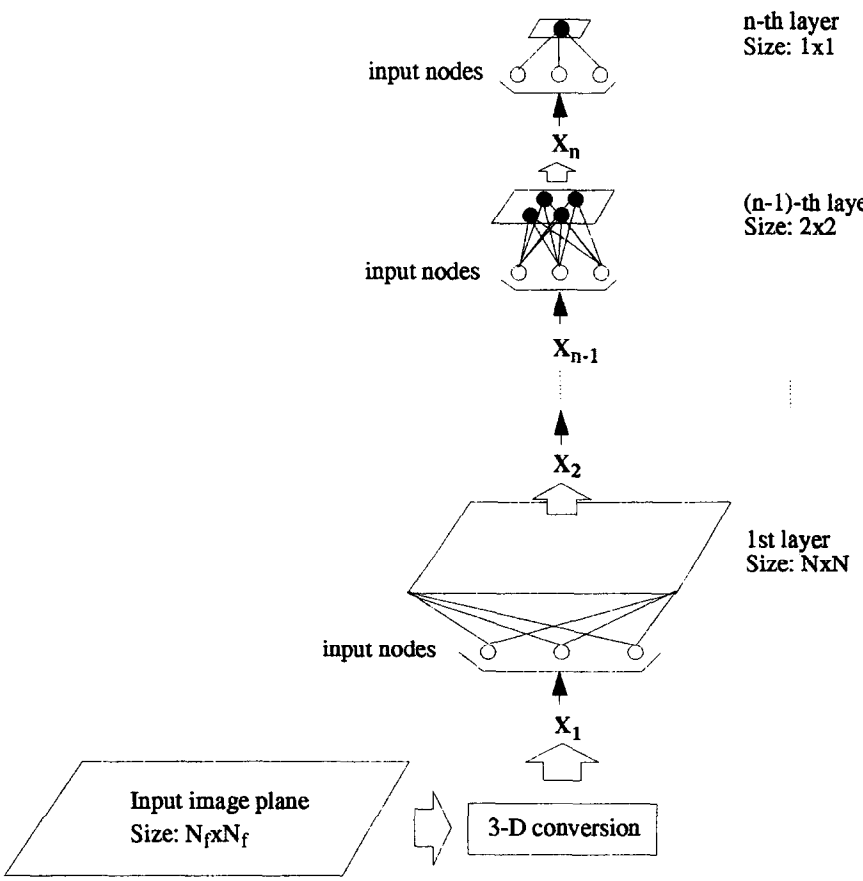


Fig. 3. The structure of the hierarchical SOM.

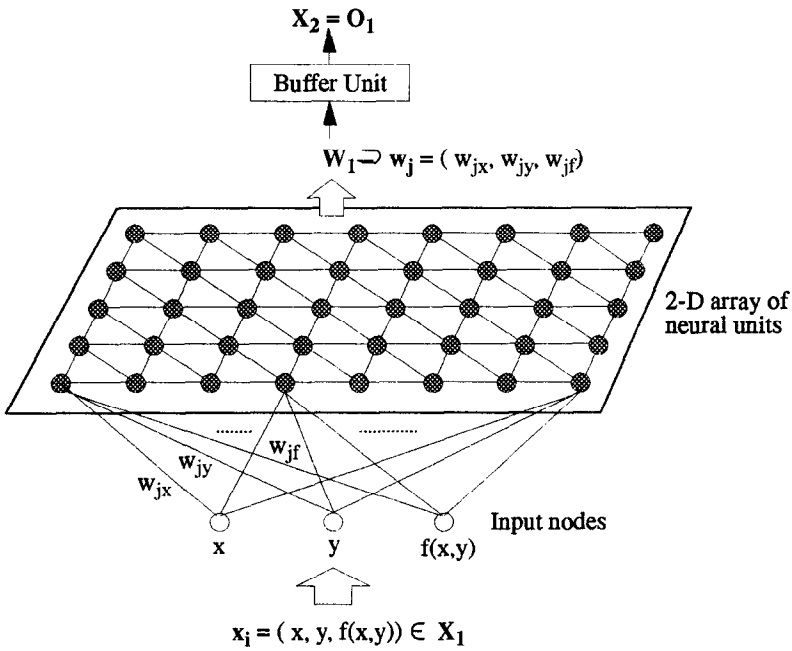


Fig. 4. A typical competitive layer (the first layer) in the HSOM.

to all the neural units in the first layer. On completion of competitive learning in the first layer, the resulting weight vectors are converted and propagated to the next layer as input to that layer. The above process is repeated until the top layer in the HSOM is reached. Without loss of generality, one may assume that the input to the HSOM is a square image of size $N_f \times N_f$, and the neural units in each competitive layer are organized as square arrays. One may further assume that the size of competitive layers are 1×1, 2×2, 4×4, ..., $N \times N$, where $N = 2^n = N_f/2$ is the size of the first layer. This structure very much resembles the quadtree data structure described in [27]. The structure of the HSOM is illustrated in Fig. 3. Fig. 4 shows the structure of a typical competitive layer in the HSOM.

2.2. Computation of the input vectors

The input feature vector for every pixel in an image needs to be computed before segmentation of the image by the HSOM. In the case of intensity images the input feature vector is simply $\mathbf{x}_i = (x, y, f(x, y))$ where $f(x, y)$ is the image intensity at pixel (x, y). In the case of range images, the input feature vector is given by the 8-tuple $\mathbf{x}_i = (x, y, z, n_x, n_y, n_z, H, K)$ where (x, y) are the image coordinate values, z the range value, (n_x, n_y, n_z) the unit normal vector, and H and K the mean and Gaussian curvature values, respectively. The surface normal and curvature values can be directly computed from a range image using window-based operators. These operators are implemented as fixed-weight feed-forward neural networks as described below.

2.2.1. Computation of the normal vector

The unit normal vector, $\mathbf{n}_p$, at a surface point $\mathbf{p}$, is perpendicular to the tangent plane at that point. The unit normal vector, $\mathbf{n}_p$ can be computed as the normalized cross-product of the two first-order derivatives of a range vector with respect to the two orthogonal $\mathbf{u}$ and $\mathbf{v}$ axes:

$$\mathbf{n}_p = \frac{(\partial \mathbf{p}/\partial u) \times (\partial \mathbf{p}/\partial v)}{\|(\partial \mathbf{p}/\partial u) \times (\partial \mathbf{p}/\partial v)\|}. \tag{1}$$

The first-order derivatives are estimated by two 5×5 equally weighted least squares derivative estimation operators, D_u and D_v, given by $D_u = \mathbf{d}_0 \cdot \mathbf{d}_1^T$ and $D_v = \mathbf{d}_1 \cdot \mathbf{d}_0^T$ where $\mathbf{d}_0 = (1/5)[1,1,1,1,1]^T$ and $\mathbf{d}_1 = (1/5)[-2,-1,0,1,2]^T$.

2.2.2. Computation of surface curvatures

The surface curvature is estimated by approximating the derivative of the surface normal in a local neighborhood [11]. A simple estimate of the surface curvature at pixel $\mathbf{p}$ in the direction of pixel $\mathbf{q}$ is given by:

$$k(\mathbf{p},\mathbf{q}) = \frac{\|\mathbf{n}_p - \mathbf{n}_q\|}{\|\mathbf{p} - \mathbf{q}\|} \times s(\mathbf{p},\mathbf{q}), \tag{2}$$

where $s(\mathbf{p},\mathbf{q}) = 1$ if $\|\mathbf{p} - \mathbf{q}\| \leq \|\mathbf{n}_p - \mathbf{n}_q\|$ and $s(\mathbf{p},\mathbf{q}) = -1$ otherwise. Here, $\mathbf{n}_p$ and $\mathbf{n}_q$ are the unit normal vectors at points $\mathbf{p}$ and $\mathbf{q}$, respectively. Eq. (2) makes two assumptions:

1. the underlying surface is smooth, and
2. $\mathbf{q}$ is close enough to $\mathbf{p}$ on the surface so that the arc length can be adequately approximated by the Euclidean distance $\|\mathbf{p} - \mathbf{q}\|$.

The sign factor $s(\mathbf{p},\mathbf{q})$ states that if the two surface normals, $\mathbf{n}_p$ and $\mathbf{n}_q$ at pixels $\mathbf{p}$ and $\mathbf{q}$ respectively, approach each other as $\mathbf{q}$ approaches $\mathbf{p}$ then the surface curvature has a negative value indicating a concave surface, else, the surface curvature has a positive value indicating a convex surface. Using the value of $k(\mathbf{p},\mathbf{q})$ one can compute the mean and Gaussian curvatures. Let $\Omega(\mathbf{p})$ be the set of pixels in the neighborhood of pixel $\mathbf{p}$. The mean curvature, H and Gaussian curvature, K are given by:

$$H = \frac{\left(k^{\min}(\mathbf{p}) + k^{\max}(\mathbf{p})\right)}{2}, \tag{3}$$

$$K = \left(k^{\min}(\mathbf{p}) \times k^{\max}(\mathbf{p})\right), \tag{4}$$

where $k^{\min}(\mathbf{p}) = k(\mathbf{p},\mathbf{q}_0) = \min_{\mathbf{q} \in \Omega(\mathbf{p})} k(\mathbf{p},\mathbf{q})$ and $k^{\max}(\mathbf{p}) = k(\mathbf{p},\mathbf{q}_1) = \max_{\mathbf{q} \in \Omega(\mathbf{p})} k(\mathbf{p},\mathbf{q})$. A 3×3 neighborhood is used when computing the mean and Gaussian curvatures.

We use a combination of MAXNETs, MINNETs and fixed-weight networks as shown in Fig. 5 to compute the values of the mean and Gaussian curvatures. As input, each pixel provides it's position vector and the unit normal vector which is computed using Eq. (1). Each curvature computing unit receives these inputs from a center pixel and its 8 nearest neighbors, and computes a curvature value. The MAXNET and

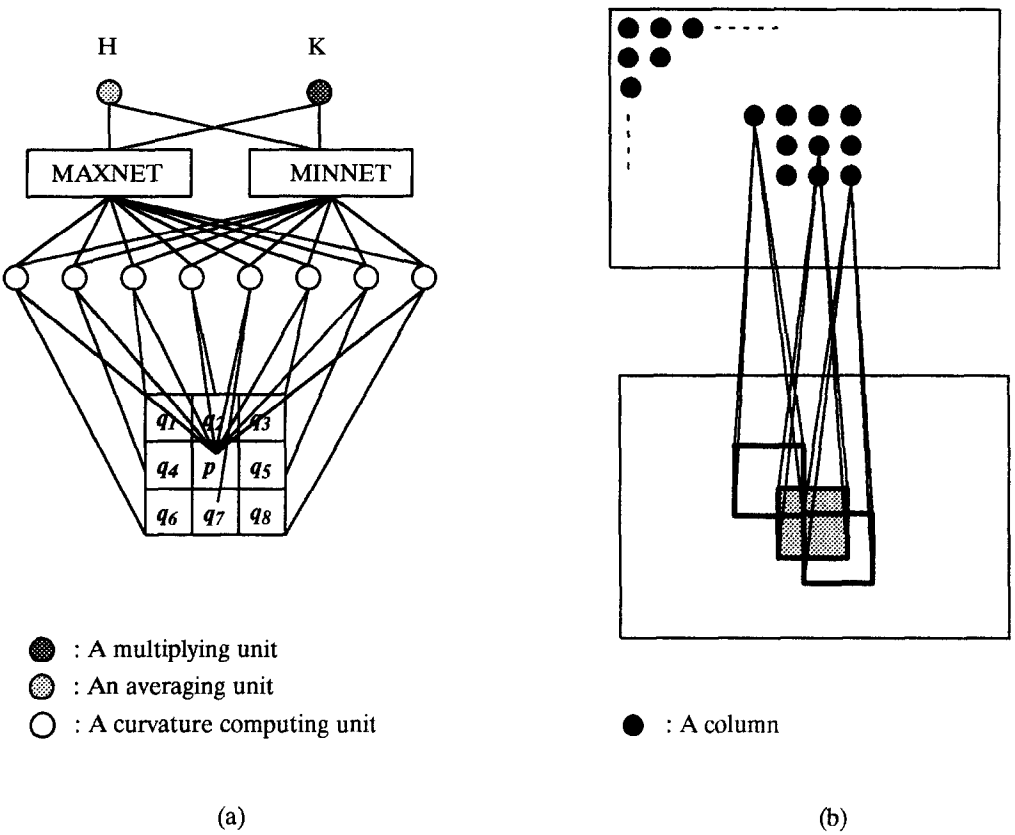


Fig. 5. Neural network implementation for computation of H and K.

MINNET collect the curvature values to find a maximum and a minimum curvature value respectively. The last units (i.e. the multiplying unit and the averaging unit) compute the H and K values respectively using the maximum and minimum curvature values from the MAXNET and MINNET respectively. We refer to this structure for computing the H and K values as an $H\,K$ column. Each $H\,K$ column receives input from a 3×3 window defined over the input image. The number of $H\,K$ columns equals the number of pixels in an input image. The overlapping windows (Fig. 5(b)) constitute the predefined receptive fields of the $H\,K$ columns. It is to be noted that the preprocessing involves only estimating the values of H and K at a given pixel location in the range image. The range image pixel is not classified as belonging to a particular qualitative surface type (i.e. spherical, cylindrical, etc.) based on the values of H and K as is commonly done in most range image segmentation algorithms [3,11].

2.3. Flow of information in the HSOM

Let $\mathbf{x}_i$ be an input vector where $\mathbf{x}_i = (x, y, f(x, y))$ for an intensity image and $\mathbf{x}_i = (x, y, z, n_x, n_y, n_z, H, K)$ for a range image as discussed earlier. The set of input

vectors $\mathbf{X}_1 = \{\mathbf{x}_i;\ i = 1,2,\ldots,N_f^2\}$ constitutes the input space for the first layer in the HSOM. Let $\mathbf{W}_1$ be the set of the resulting weight vectors from the first layer, where $\mathbf{W}_1 = \mathbf{w}_j = \{(w_x, w_y, w_f),\ j = 1,2,\ldots,N^2\}$ for intensity images and $\mathbf{W}_1 = \{\mathbf{w}_j = (w_x, w_y, w_z, w_{n_x}, w_{n_y}, w_{n_z}, w_H, w_K),\ j = 1,2,\ldots,N^2\}$ for range images. On completion of training, $\mathbf{W}_1$ contains the representation vectors corresponding to the first level of abstraction of the input vectors. Let $\mathbf{O}_1 \subseteq \mathbf{W}_1$ be the set of weight vectors that correspond only to the winning neural units in the first layer. $\mathbf{O}_1$ is used as the input vector set for the second layer in the HSOM, i.e. $\mathbf{X}_2 = \mathbf{O}_1$. Only the weight vectors of the winning units in a previous layer are used as the input to the next layer since they alone are deemed to be the correct representation vectors. Other neural units are adapted to interpolate the values of winning units. This follows directly from the ordering property of the SOM as shown by Kohonen for the one-dimensional case [15].

The interpretation of $\mathbf{X}_i$, $\mathbf{W}_i$, $\mathbf{O}_i$, $i = 2,3,\ldots,n$ is straightforward. Note that the sizes of $\mathbf{W}_i$ are fixed, but the sizes of $\mathbf{X}_i$ and $\mathbf{O}_i$ are not determined a priori since the number of winning neural units is dependent on the input to that layer. Let TM be the topological mapping performed by a single layer SOM: $\mathrm{TM}: \mathbf{X} \mapsto \mathbf{O}$, whose domain is $\mathbf{X}_i$ and range is $\mathbf{O}_i$. For an n-layer HSOM, the input $\mathbf{X}_1$ is mapped to $\mathbf{O}_n$ by TM^n, given by:

$$\mathrm{TM}: \mathbf{X}_i \mapsto \mathbf{O}_i, \quad i = 1,2\ldots,n, \tag{5}$$

$$\mathrm{TM}^n: \mathbf{X}_1 \mapsto \mathbf{O}_1 = \mathbf{X}_2 \mapsto \mathbf{O}_2 = \mathbf{X}_3 \mapsto \ldots \mapsto \mathbf{O}_{n-1} = \mathbf{X}_n \mapsto \mathbf{O}_n, \tag{6}$$

$$\mathrm{TM}^n: \mathbf{X}_1 \mapsto \mathbf{O}_n. \tag{7}$$

2.4. The learning procedure in the HSOM

The learning procedure in each layer of the HSOM consists of the same three steps as the learning procedure in the single-layer SOM described in Section 1.4. The competitive phase of the learning process (comprising of the first two steps) determines the winning neural unit for weight adjustment. The cooperative phase (comprising of the third step) adjusts the weight vectors. We modified the original learning algorithm proposed by Kohonen [15] by incorporating a scheme proposed by Ritter and Schulten [25] for adapting the parameters for weight adjustment in the cooperative phase of the learning algorithm.

2.4.1. Weight vector adjustment

After a winning neural unit is determined, the weight vectors of the neural units in its neighborhood are adjusted as follows:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + lr(t) \cdot h_{rs}(t) \cdot \big(\mathbf{x}(t) - \mathbf{w}_j(t)\big), \tag{8}$$

$$lr(t) = lr_{\text{init}} \cdot \left(\frac{lr_{\text{final}}}{lr_{\text{init}}}\right)^{t/t_{\max}}, \tag{9}$$

$$h_{rs}(t) = \exp\left(-\frac{\|\mathbf{i} - \mathbf{j}\|^2}{2 \cdot \sigma^2(t)}\right), \tag{10}$$

$$\sigma(t) = \sigma_{\text{init}} \cdot \left(\frac{\sigma_{\text{final}}}{\sigma_{\text{init}}} \right)^{t/t_{\max}}, \tag{11}$$

where **i** is the location vector of a winning neural unit, **j** is the location vector of a neighboring neural unit, and $lr(t)$ is the learning rate.

In our computer simulations of the HSOM the parameters in the above equations were set empirically. The learning rate $lr(t)$ was decreased from $lr_{\text{init}} = 0.8$ to $lr_{\text{final}} = 10^{-3}$ in order to ensure final convergence to the asymptotic values. In an HSOM layer of size $N \times N$, the parameters, σ_{init}, σ_{final}, and $t_{\max}$ were set as follows: The value of σ_{init} was set to $N_{\max}/2$ and that of σ_{final} to 0.1 where $N_{\max}$ is the maximum neighborhood radius. Most researchers have used and recommended a maximum initial neighborhood size of $(N \times N)/4$. Accordingly, we set the value of $N_{\max} = N/2$. With each iteration of the learning procedure, the lateral width h_{rs}, which redefines the neighborhood range, was slowly decreased to a single winning neural unit in order to localize the sensitivities of the neural units. The same learning procedure was used in every layer. The initial weight vectors were set to random values with a small variance around the average of the input vectors. In each competitive layer, the two phases of learning were applied repeatedly a sufficient number of times to ensure that the computed error for all input vectors fell within an acceptable range. The value of $t_{\max}$, which was experimentally determined, was selected to be large enough to ensure that the computed error fell within the acceptable error range for all the intensity images. Accordingly, the value of $t_{\max}$ was set to $100 \times (N_{\max} + 1)$.

2.4.2. Distance measure

In the case of intensity images, the real distance measure, i.e. the Euclidean norm between a weight vector and an input vector, was taken to reflect the error. In the case of range image data, there are four heterogeneous pieces of information in a feature vector – the position vector, (x,y,z), the unit normal vector, (n_x,n_y,n_z), and the curvature values, H and K. To fine-tune the extent of the contributions from each of these four components of information, the Euclidean distance measure used in the case of intensity images was modified. A *weighted Euclidean distance measure* which is a weighted sum of four separate Euclidean distance measures: one between the position vectors, one between the unit normal vectors, and one each from the two curvature values, was defined thus:

$$d(\mathbf{x}_i,\mathbf{w}_j) = a \cdot \|\mathbf{x}_{\mathbf{p}} - \mathbf{w}_{\mathbf{p}}\| + b \cdot \|\mathbf{x}_{\mathbf{n}} - \mathbf{w}_{\mathbf{n}}\| + c \cdot |H - w_H| + d \cdot |k - w_K|, \tag{12}$$

where $0 \le a,b,c,d \le 1$, $\mathbf{x}_i = (x,y,z,n_x,n_y,n_z,H,K) = (\mathbf{x}_{\mathbf{p}};\mathbf{x}_{\mathbf{n}};H;K)$, $\mathbf{x}_{\mathbf{p}} = (x,y,z)$, $\mathbf{x}_{\mathbf{n}} = (n_x,n_y,n_z)$, $\mathbf{w}_i = (w_x,w_y,w_z,w_{n_x},w_{n_y},w_{n_z},w_H,w_K) = (\mathbf{w}_{\mathbf{p}};\mathbf{w}_{\mathbf{n}};w_H;w_K)$, $\mathbf{w}_{\mathbf{p}} = (w_x,w_y,w_z)$ and $\mathbf{w}_{\mathbf{n}} = (w_{n_x},w_{n_y},w_{n_z})$. The weighted Euclidean distance measure, permits one to selectively control the relative dominance of the contribution of each component. The actual weights used in the computer simulations are described in Section 3. The same set of values for $\{a,b,c,d\}$ were used for all the range images in the computer simulations.

2.5. Neighborhood topology and topology preserving mapping

A major difference between the SOM and other CLNNs is that associated with each neural unit in the SOM there is a topological neighborhood and all the neural units in the

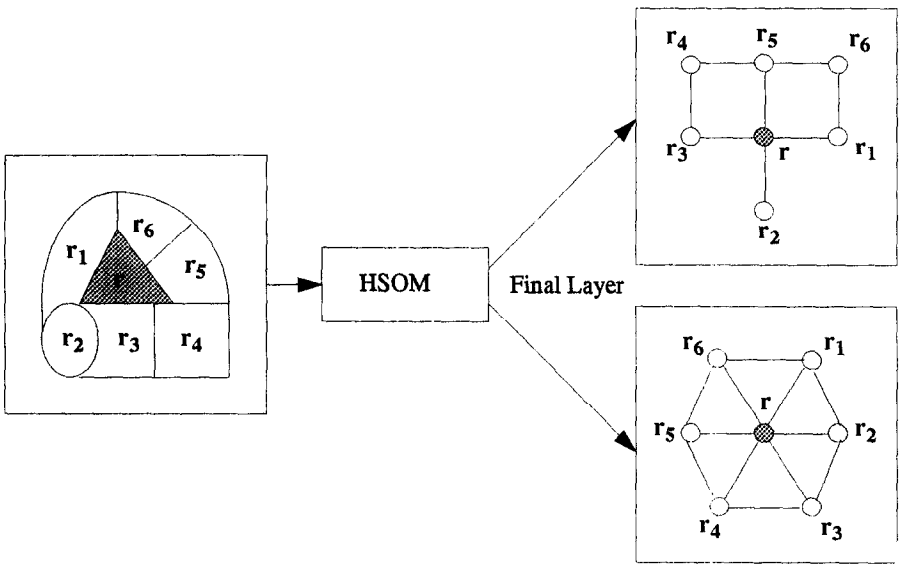


Fig. 6. The effect of neighborhood topology on mapping.

neighborhood of a winning unit are updated in conjunction with the winning unit. The idea of associating a topological neighborhood with each neural unit in the SOM has its origins in the study of the lateral interaction between cells of the two-dimensional neural layers in the mammalian brain [15]. It is precisely due to the fact that all the neural units in the topological neighborhood of the winning unit are updated along with the winning unit that the SOM exhibits a topological ordering property. A simple proof of this property appears in [15]. An appropriate definition of a topological neighborhood is therefore crucial for the SOM to function as a topology preserving (i.e. topographic) mapping.

The topological structure of the neighborhood of the neural units determines the dimension of the topological ordering of input vectors. In general, for a single-layer SOM, either 4-, 6- or 8-neighbors could be used. A suitable definition of the neighborhood topology is very important if a single-layer SOM is to be used for image segmentation. The neighborhood topology determines the final topology of the mapping and therefore the shape of the regions in the resulting segmented image. The topological neighborhood relationship in the original single-layer SOM is embedded *entirely* in the interconnection topology of the neural units. If two neural units share a connection, then the corresponding regions mapped onto those two neural units are adjacent. For example, consider six regions $\mathbf{r}_i$, $i = 1,2,\ldots,6$, neighboring a region $\mathbf{r}$ as shown in Fig. 6. The topological mapping cannot be preserved by a single mapping if we choose the four-nearest-neighbors neighborhood topology, since $\mathbf{r}_4$ and $\mathbf{r}_6$ are not mapped as neighbors of $\mathbf{r}$ (Fig. 6). But the use of the hexagonal neighborhood topology, for this particular example, would map all six regions onto neural units which are direct

neighbors of the neural unit corresponding to **r**, thus resulting in a topologically correct mapping.

The HSOM, like the conventional single-layer SOM, is a topology preserving mapping. In any single layer of the HSOM, the topological neighborhood relationship is dependent on the interconnection topology of the neural units just as in the case of the single-layer SOM. However, the choice of the neighborhood topology is not as critical for the HSOM as it is for the SOM. Also, the number of neural units in a single layer of the HSOM is not as crucial as in the case of the single-layer SOM.

The above characteristics of the HSOM could be attributed to the fact that some of the neighborhood information is encoded by the abstraction tree in the sense that the child nodes of the same parent node have a higher probability of being topological neighbors. The only requirement is that the number of layers in the HSOM is adequate to segment the most complex region in the image. This requirement is satisfied if the number of neural units in the first layer of the HSOM is greater than or equal to the number of visually distinguishable regions in the image. In our case the number of neural units in the first competitive layer of the HSOM is $N \times N$ where $N = N_f/2$ for an input image of size $N_f \times N_f$. It is quite clear that the number of neural units in the first competitive layer of the HSOM (i.e. $N^2 = N_f^2/4$) is more than adequate for most images of size $N_f \times N_f$. In other words, it would be very difficult to imagine an image of size $N_f \times N_f$ in which the number of distinguishable regions exceeded $N^2 = N_f^2/4$. However, it may be possible to optimize the number of neural units in the first layer and consequently the total number of layers in the HSOM depending on the input image complexity. This remains a topic for future research.

In short, the HSOM is less sensitive to the choice of neighborhood topology and the number of neural units in a given layer of the HSOM than the single-layer SOM in the context of image segmentation. For the computer simulations of the HSOM described in the following section, a circular neighborhood was used where the size of the neighborhood was made as a function of $((i-k)^2 + (j-l)^2)^{1/2}$ where (i,j) and (k,l) are the positions of the neural units in the competitive layer.

One of the significant advantages of the HSOM is that one can also consider different resolutions of topological ordering. The size of the regions corresponding to the input vectors on which the topological ordering is defined is different for each layer of the HSOM. A neural unit at a higher layer represents a coarser region, i.e. the size of the region represented by that neural unit is larger. Thus, the size of the regions represented by the vectors mapped to a neural unit belonging to the closed set defined by the topological ordering relation varies with each layer. This is illustrated in Fig. 7. Fig. 7(a) shows examples of a topologically correct mapping. In the first layer both region a and region d are each adjacent to both, region b and region c and conversely. Two regions are considered adjacent if they share a common boundary. The regions a, b, c and d can be mapped in a topologically correct manner in two ways as depicted in the second layer. These regions are mapped to four groups of neural units which are direct neighbors thus preserving the topological relationship in the first layer. In the third layer these four regions are finally mapped to four neural units which have the same neighborhood relationship as do the four regions in the first layer. Fig. 7(b) shows the different resolutions of topological ordering. The four neural units represented by the

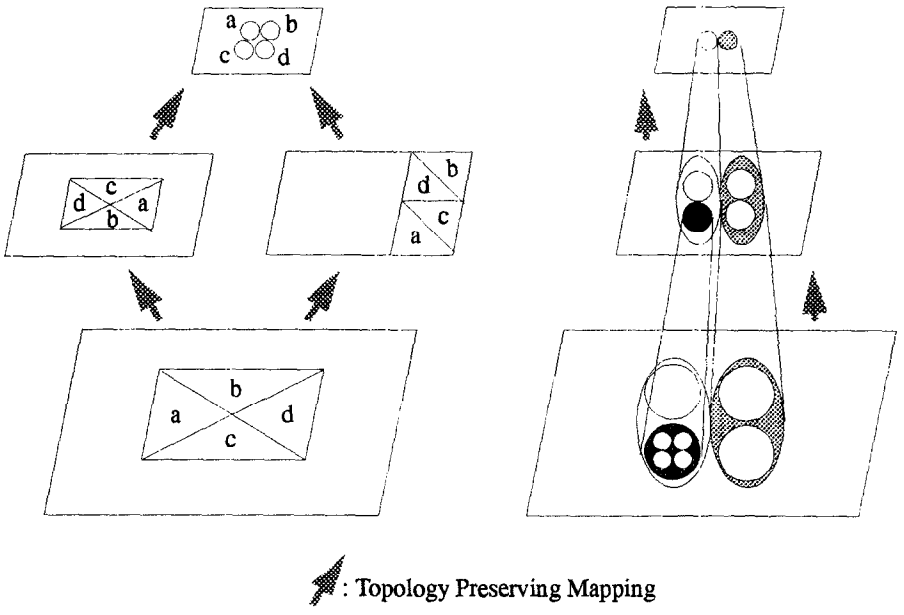


Fig. 7. Topological ordering over variable sizes of elements in each layer. (a) Maintaining a topological neighborhood, (b) Resolution of topological ordering.

four smallest circles in the first layer are mapped to a single neural unit in the second layer. Since the four neural units share the same parent node in the abstraction tree, they are considered adjacent. The four neural units in the second layer each of which represent four neural units in the first layer, are topological neighbors. The two groups of two neural units each in the second layer are mapped onto two neural units in the third layer which are topological neighbors. Thus, different resolutions of topological ordering can be explored in each layer. The four neural units in the second layer preserve the neighborhood relationship of the four medium-size circles whereas the two neural units in the third layer preserve the neighborhood relationship of the two largest circles.

3. Experimental results

The HSOM was simulated on the Connection Machine (CM-2) [1] which is a fine-grained SIMD parallel computer with up to 64K (i.e. 65,536) processors [10]. The speed-up achievable by the massively parallel computer is limited since the SOM learning procedure is inherently sequential, i.e. based on examining one input vector at a

[1] The Connection Machine (CM) is a trademark of Thinking Machines Inc., Cambridge, MA.

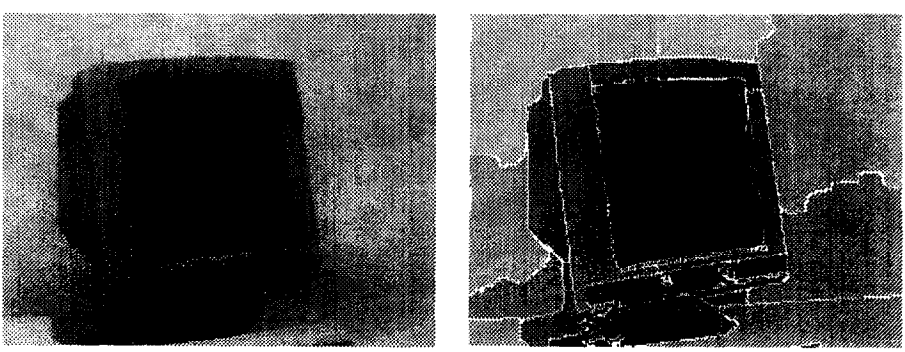
time. On completion of learning in a given layer, the input for the next layer is extracted from the winning neural units. The geometry of the CM processors is changed to that of the next layer and the learning process is repeated. For further details on the implementation of the SOM learning procedure on the CM-2, the interested reader is referred to [14,30]. The HSOM was simulated and tested on several intensity and range images. The segmentation results for three intensity images and four range images are presented in this paper.

3.1. Experimental results for intensity images

For the segmentation of intensity images, a six-layer HSOM was used in which the first layer was a neural unit array of size 32×32, the second layer 16×16, the third layer 8×8, the fourth layer 4×4, the fifth layer 2×2, and the top layer consisted of a single neural unit. The segmented image was generated by traversing the resulting abstraction tree in a breadth-first manner starting from the root node. A node in the abstraction tree was expanded if the variance of the intensity values of the pixels in the region exceeded a predefined threshold. An unexpandable node was labeled as a closed node. The closed nodes constitute the leaf nodes in the abstraction tree.

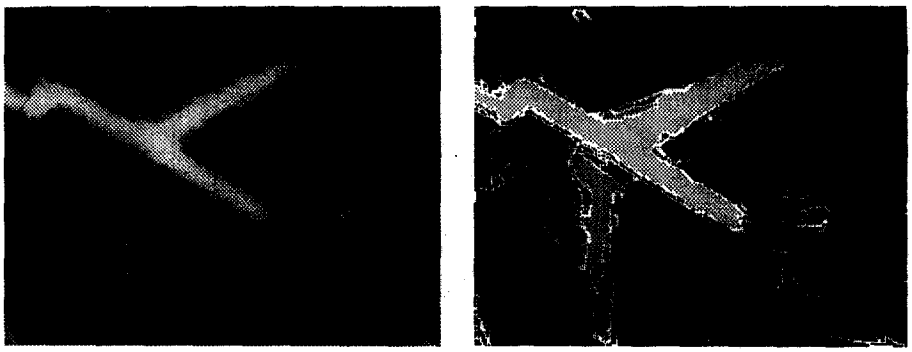
The resulting segmented image usually contains regions at different levels in the abstraction tree. This was true for the intensity images that were used in the computer simulations. Note, that for many applications, an entire abstraction tree providing a relational description may itself be considered the final outcome of the segmentation process. The entire abstraction tree may be directly propagated as input to the high-level vision processes instead of a single segmented image.

The first intensity image termed as *Monitor*, is shown in Fig. 8(a). Fig. 8(b) shows the resulting segmented image containing 32 regions, which consist of 10 regions from the fourth layer and 22 regions from the third layer. Fig. 9 and Fig. 10 show the second intensity image, *Boeing*, and the third intensity image, *Phone*, respectively, along with the resulting segmented images. In Table 1 we have tabulated the total number of visually distinguishable regions in each segmented image corresponding to the input



(a) (b)

Fig. 8. Image segmentation result for the *Monitor* image. (a) Original image, (b) Segmented image.



(a) (b)

Fig. 9. Image segmentation result for the *Boeing* image. (a) Original image, (b) Segmented image.

images *Monitor*, *Boeing* and *Phone*. Table 1 also tabulates the number of regions resulting from each level in the abstraction tree generated by the HSOM. Recall that the *closed* nodes resulting from the breadth-first traversal or expansion of the abstraction tree correspond to the visually distinguishable regions in the final segmented image. In effect, Table 1 tabulates the number of closed nodes (i.e. leaf nodes) at each level of the abstraction tree resulting from its breadth-first expansion.

3.2. Experimental results for range images

The HSOM used for the segmentation of range images was identical to the one used for the segmentation of intensity images. The segmented range image was generated by

(a) (b)

Fig. 10. Image segmentation for the *Phone* image. (a) Original image, (b) Segmented image.

Table 1
Number of segmented regions in each layer of the abstraction tree

Real Image	No. of Regions				
	Total	5-th layer	4-th layer	3rd layer	2nd layer
Monitor	47	0	10	37	0
Boeing	81	2	22	57	0
Phone	134	2	18	63	51

a breadth-first traversal of the resulting abstraction tree. In the case of range images, a node was expanded if

1. the sum of the variances of the range values, mean curvature magnitude values and the unit normal vectors of the pixels in the region was larger than a predefined threshold value, or
2. the individual variances of the range values or the mean curvature magnitude values or the unit normal vector values exceeded their corresponding predefined threshold values.

The weights used in the weighted Euclidean distance measure were determined empirically. During the course of computer simulation of the HSOM, the objects in the input images were observed to have relatively simple surfaces so that the value of the Gaussian curvature was not critical. Consequently, the contribution of the Gaussian curvature to the distance measure was de-emphasized, i.e. the weight d in the weighted Euclidean distance measure (Eq. (12)) was chosen to be very small. The magnitude of the mean curvature was found to be much more effective than the mean curvature value itself. So, the magnitude of the mean curvature was used instead of the mean curvature value. The weight ratio $a:b:c:d = 1:10^{-2}:10^{-1}:10^{-5}$ was used in the weighted Euclidean distance measure in the case of all the range images that we used in our computer simulation of the HSOM.

In order to illustrate the segmentation process as it proceeds in each successive layer of the HSOM, the intermediate results of the range image *Column* are shown in Fig. 11. The first image shows the input range image and the rest of the images illustrate the intermediate segmentation results for each successive layer in the HSOM starting with the first layer. For convenience of display, the range values are converted to intensity values between 0 and 255. Also, the segmentation results from each of the layers are mapped back to the pixel locations in the input image in a manner such that the image pixels mapped to a common neural unit on completion of learning, form a region enclosed by white boundaries.

In the case of range images, it was observed that the segmentation of planar surfaces could be accomplished easily in a few HSOM layers since the clustering is determined largely by the position vector term in the weighted Euclidean distance measure. It was observed that cylindrical surfaces were typically divided into longitudinal strips parallel to the cylinder axis at the lowest layer of abstraction in the HSOM. These strips were

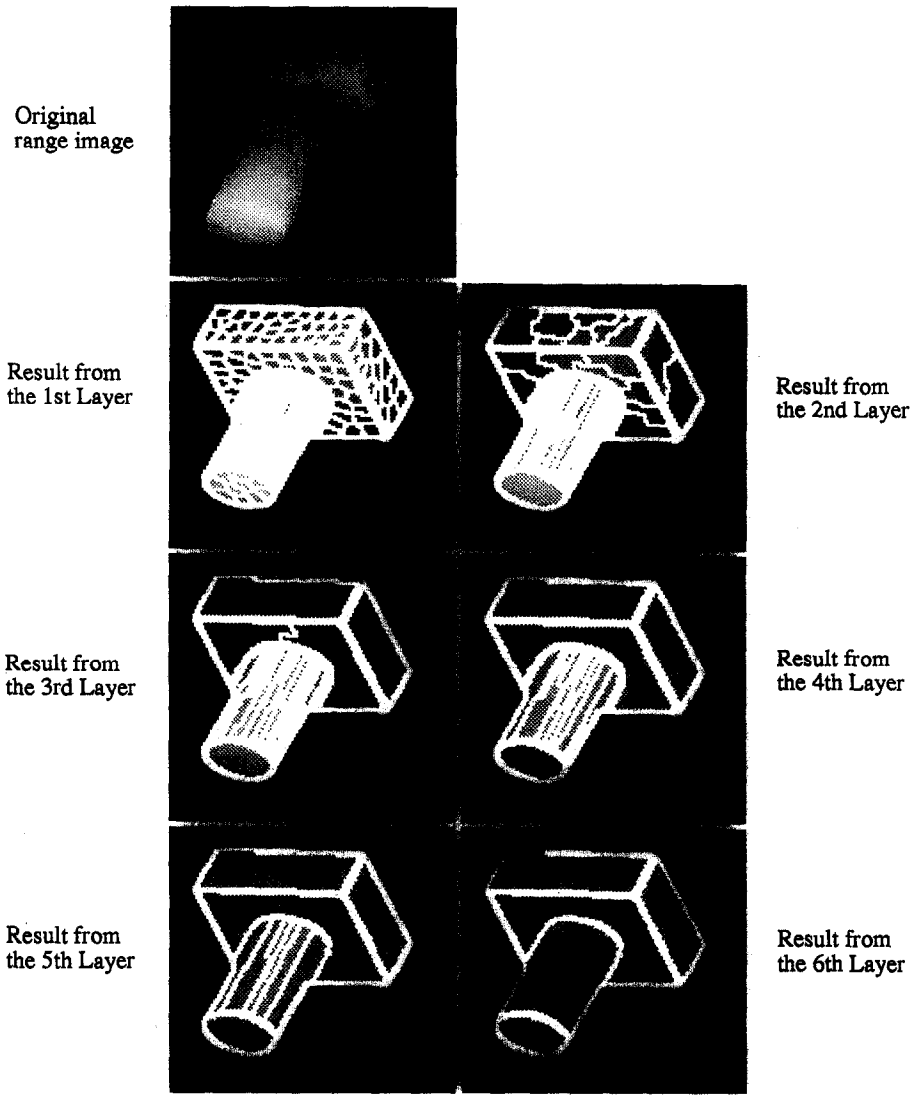


Fig. 11. The segmentation results from six layers of the HSOM for the *Column* image.

successively merged into larger surface patches as one proceeded to higher layers of abstraction (Fig. 11). It was also observed that conical surfaces were typically divided into fan-like regions at the lowest layer of abstraction in the HSOM. These fan-like regions were then merged in successive layers of the HSOM. The reason for cylindrical surfaces to be divided into longitudinal strips and for conical surfaces to be divided into fan-like segments is that the points along the contours of these strips or segments have the same (or similar) values for the surface normal. The relatively larger weight assigned to the mean curvature magnitude term in comparison to the weight assigned to the unit

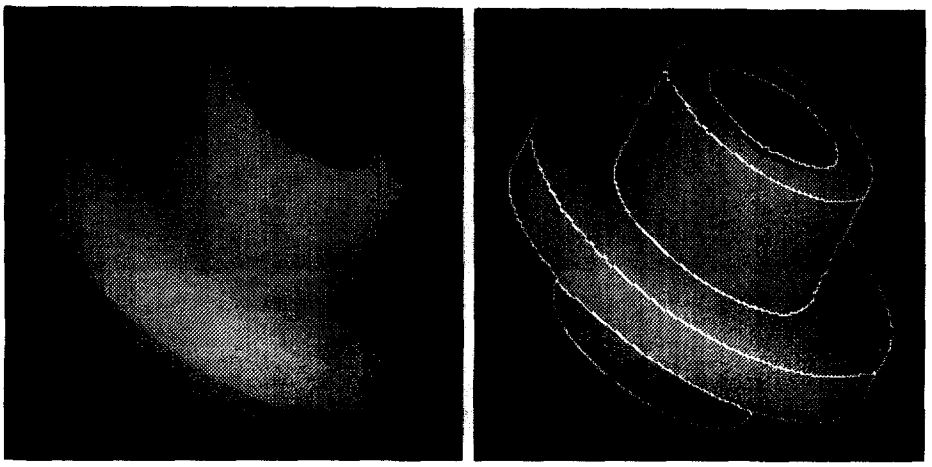


Fig. 12. (a) Input range image *Agpart*, (b) Segmentation result.

normal vector term in the weighted Euclidean measure as well as the hierarchical structure of the HSOM ensured smooth merging of subregions belonging to a cylindrical, conical or spherical region.

As a general observation, the number of HSOM layers needed to successfully segment a range surface is an increasing function of the complexity of underlying surface. Consequently, planar surfaces being the least complex (since all points on a planar surface have identical unit normal vectors and zero H and K values) were seen to be the easiest to segment. Unlike the segmentation of intensity images, in the case of the range images used in our computer simulations, all the closed nodes occurred at the same level of abstraction in the abstraction tree.

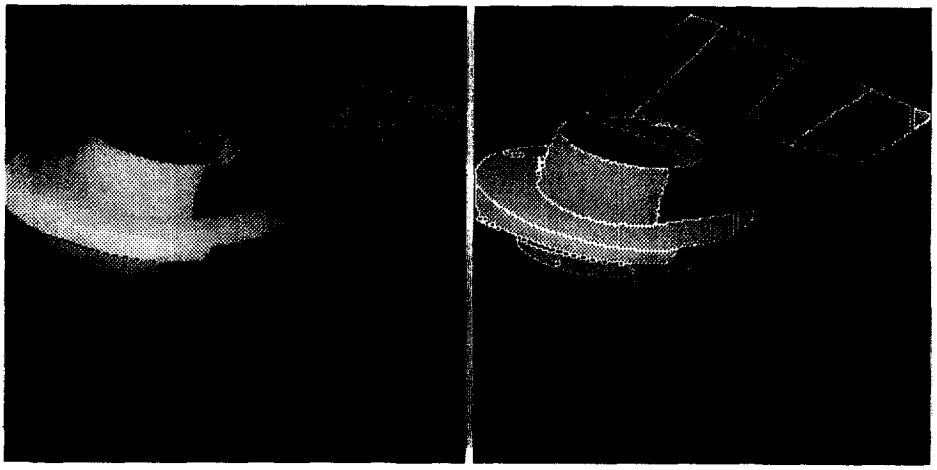


Fig. 13. (a) Input range image *Agpart + block*, (b) Segmentation result.

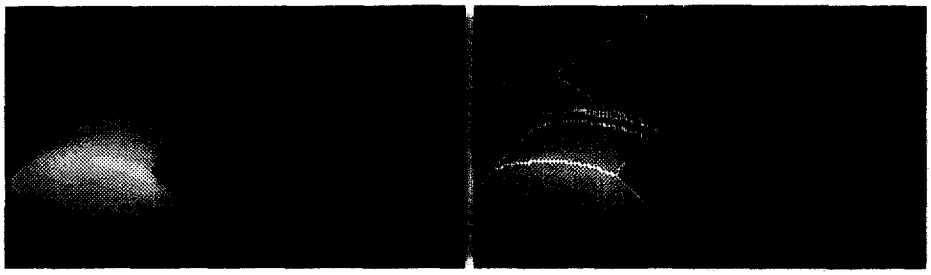


Fig. 14. (a) Input range image *Box + cap*, (b) Segmentation result.

Figs. 12–15 show four of the several range images that were used in the experiments along with the corresponding segmentation results using the HSOM. In order to delineate the boundaries between surface regions, the edges between distinct regions were marked by white lines in the image which displays the final result of the segmentation process. The final results show that the various surface regions in the scene have been well delineated. The small noise-like circular regions in the input range images and the two lines dividing the objects in the *Agpart + block* range image were due to the lack of data at certain pixel locations in the input range images (i.e. due to shadows or noise). Shadows are typical of triangulation-based range sensors and arise whenever the portions of the scene within the field of view of the camera are not illuminated by the light source.

3.3. Comparison with an edge-based segmentation technique

To bring out the advantages of the HSOM when compared to more traditional segmentation techniques, we present the results of an edge-based segmentation tech-

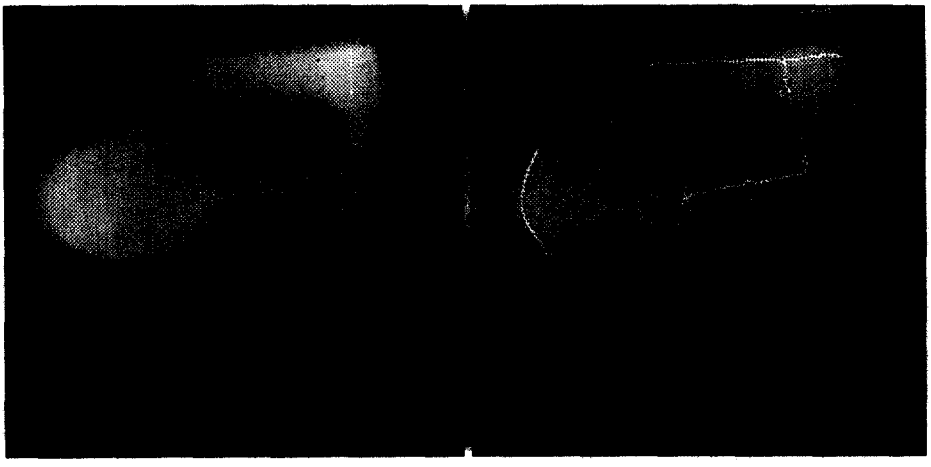


Fig. 15. (a) Input range image *Cup + block*, (b) Segmentation result.

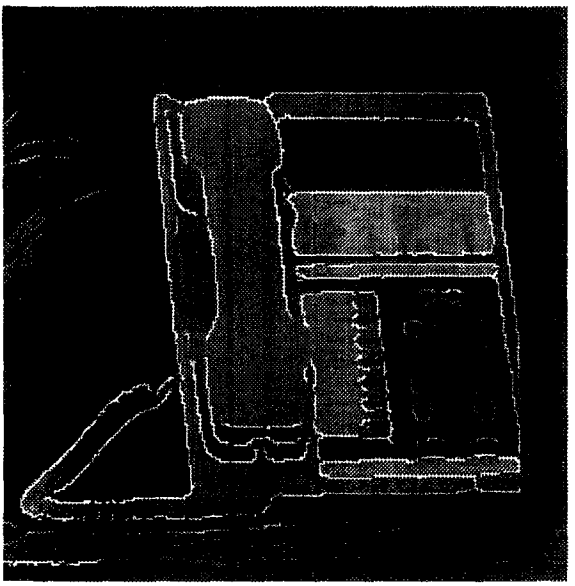


Fig. 16. Segmentation of the *Phone* image with Canny edge detector, $\sigma = 2$.

nique on the intensity images presented in Section 3.1. The intensity images *Phone*, *Boeing* and *Monitor* were convolved with a Canny edge operator [5]. The Canny edge operator essentially consists of three operations: (1) smoothing operation with a Gauss-

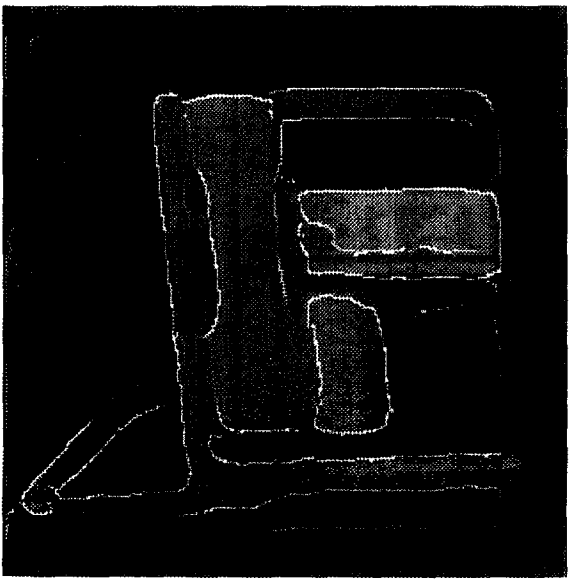


Fig. 17. Segmentation of the *Phone* image with Canny edge detector, $\sigma = 5$.

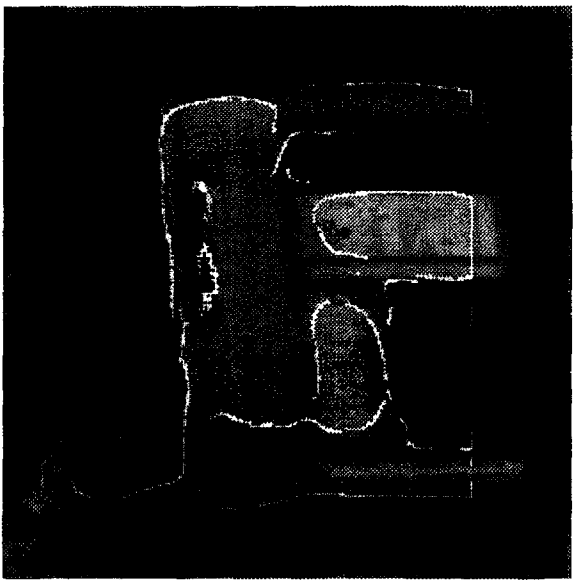


Fig. 18. Segmentation of the *Phone* image with Canny edge detector, $\sigma = 10$.

ian filter characterized by the parameter σ which is the standard deviation associated with a zero mean Gaussian distribution, (2) computation of the extrema of the directional gradient values for the smoothed image, and (3) suppression of false extrema

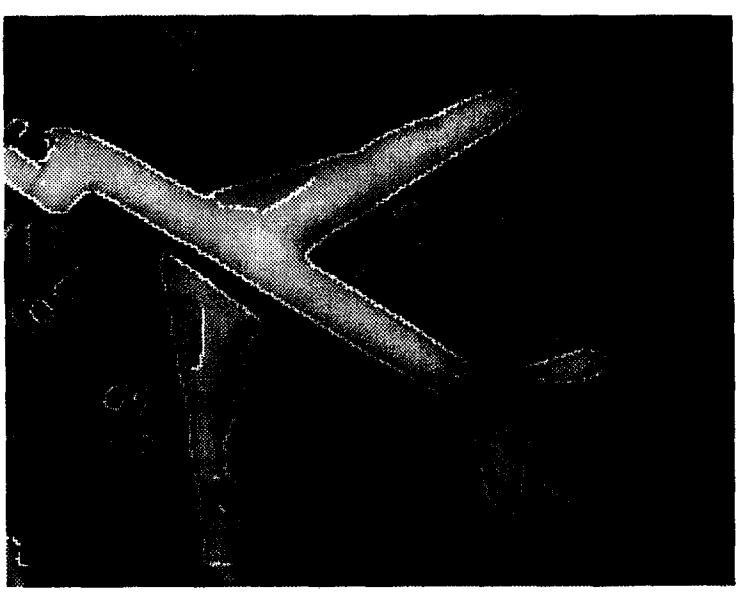


Fig. 19. Segmentation of the *Boeing* image with Canny edge detector, $\sigma = 2$.

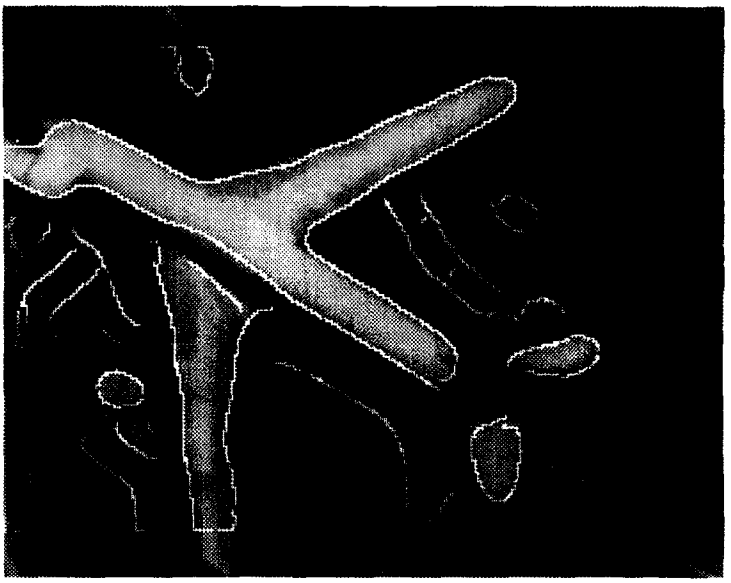


Fig. 20. Segmentation of the *Boeing* image with Canny edge detector, $\sigma = 5$.

arising from additive white noise. The extrema of the directional gradient values that survive the false extrema suppression process correspond to the edges detected at the scale σ.

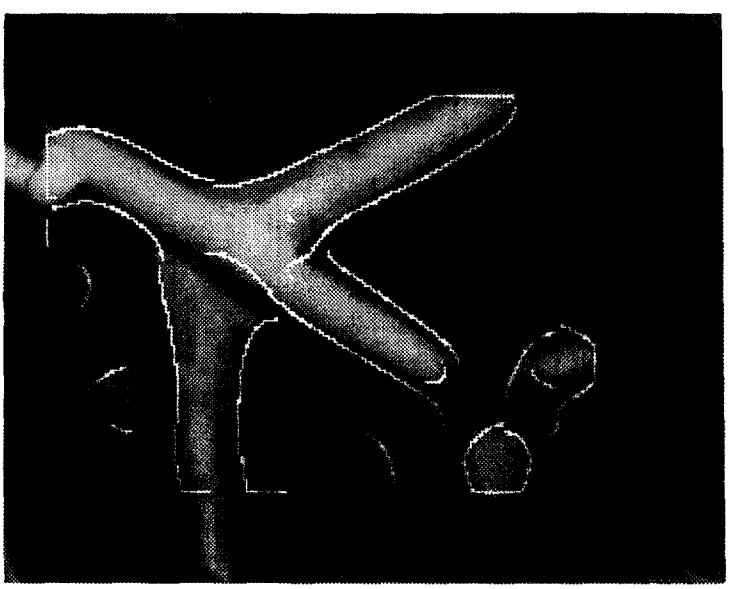


Fig. 21. Segmentation of the *Boeing* with Canny edge detector, $\sigma = 10$.

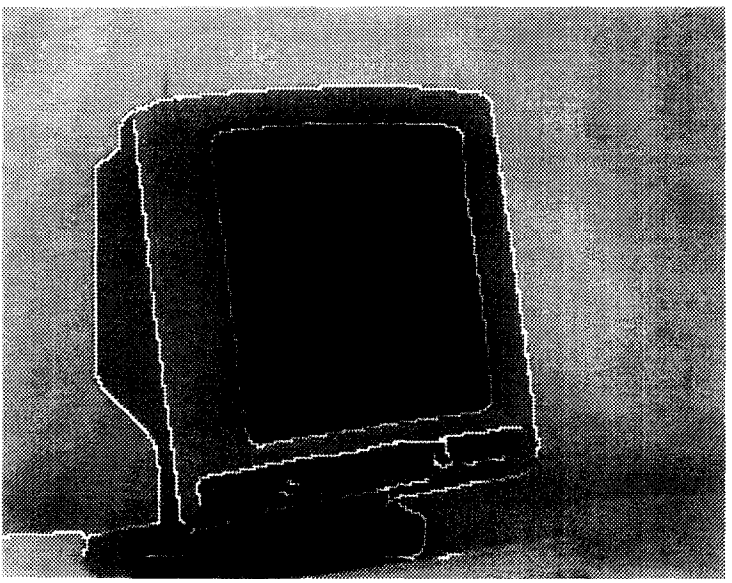


Fig. 22. Segmentation of the *Monitor* image with Canny edge detector, $\sigma = 2$.

Fig. 16, Fig. 17 and Fig. 18 show the result of overlaying the output of Canny edge detector at scales $\sigma = 2$, $\sigma = 5$ and $\sigma = 10$, respectively, on the gray scale *Phone* image. Fig. 19, Fig. 20 and Fig. 21 show the corresponding results for the gray scale

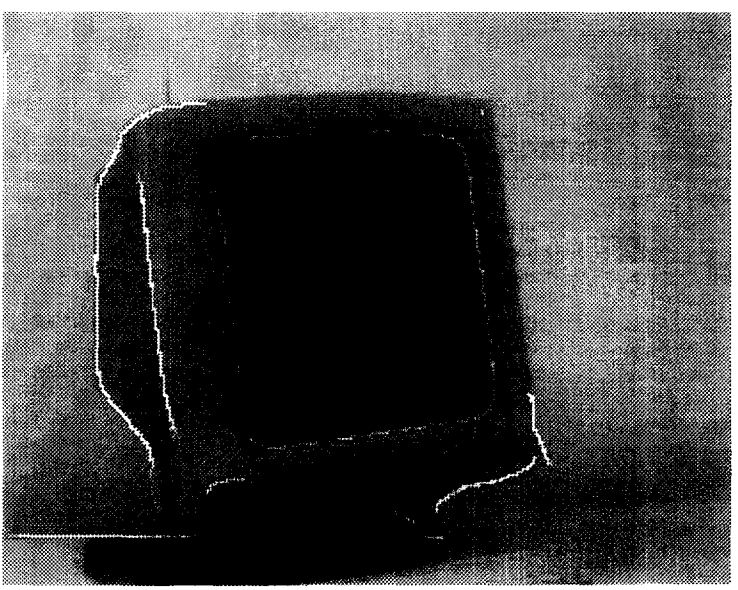


Fig. 23. Segmentation of the *Monitor* image with Canny edge detector, $\sigma = 5$.

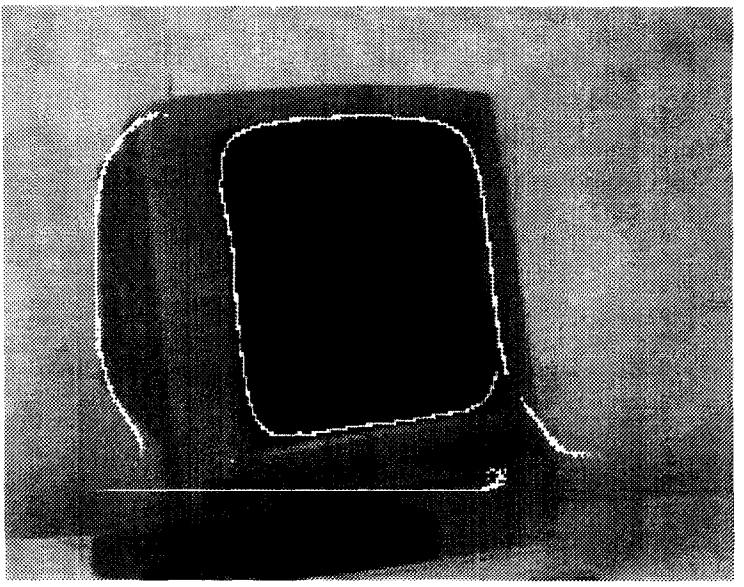


Fig. 24. Segmentation of the *Monitor* image with Canny edge detector, $\sigma = 10$.

Boeing image, whereas Fig. 22, Fig. 23 and Fig. 24 show the corresponding results for the gray scale *Monitor* image. These results bring out the limitations of segmentation at a single scale of abstraction. At lower values of σ, one can observe edges introduced by noise and unnecessary detail. One can also observe fragmentation of edges which correspond to significant object boundaries thereby resulting in region boundaries in the segmented image that are not closed. Thus, for low values of σ certain regions of the image are oversegmented whereas others are undersegmented. At higher values of σ one can observe dislocation and distortion of the edge contours which also results in incorrect segmentation.

The performance of the HSOM was also compared to that of a well-known range segmentation algorithm by Leonardis et al. [16] in terms of oversegmentation, misclassification and number of missed pixels. Leonardis' algorithm based on surface fitting was seen to suffer from oversegmentation, misclassification, and a greater number of missed pixels as compared to the HSOM. In summary, the above experiments positively demonstrated the utility of the HSOM for range and intensity image segmentation, both in concept and in practice. The HSOM also demonstrated less sensitivity to the choice of segmentation parameters.

4. Conclusions

This paper considered the use of neural networks for an important low-level computer vision problem, namely, image segmentation. Since the problem of image segmentation can be formulated as one of vector quantization, one is motivated to consider the use of

a Self-Organizing Map (SOM). However, the original single-layer SOM has certain inherent limitations in the context of image segmentation. The SOM was extended to include multiple competitive layers which were organized as a pyramidal structure. The new extended network, which we have termed as the Hierarchical Self-Organizing Map (HSOM), enables multiscale or hierarchical image segmentation, producing various levels of abstraction in the feature space. The result of the segmentation process which includes all intermediate levels of abstraction could be organized as an *abstraction tree*. Breadth-first traversal of the abstraction tree could be used to produce segmented images.

The HSOM was simulated and tested on several intensity and range images. The HSOM was also compared with an edge-based segmentation technique using the Canny edge detector. The experimental results were very satisfactory in demonstrating the superiority of the HSOM over conventional techniques for image segmentation. Although the immediate concern of this paper was image segmentation using the HSOM, the HSOM could be potentially used in any application that calls for clustering of vectors in feature space. The principal advantages of the HSOM in the general context of feature vector clustering are that the number of clusters need not be specified a priori, and that one obtains a multiscale abstraction of clusters in feature space.

Acknowledgements

The authors wish to thank the Pattern Recognition and Image Processing Laboratory, Department of Computer Science, Michigan State University, East Lansing, Michigan, for range images from their range image database. The authors wish to thank Professor William Sakoda, Department of Computer Science, State University of New York, Stony Brook, New York for making available his implementation of the Canny edge detector. The authors also wish to thank the anonymous reviewers for their insightful comments and suggestions that greatly improved the final version of the paper.

References

[1] S.C. Ahalt, A.K. Krishnamurthy, P. Chen and D.E. Melton, Competitive learning algorithms for vector quantization, *Neural Networks* 3(3) (1990) 277–290.

[2] D.H. Ballard and C.M. Brown, *Computer Vision* (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982).

[3] P.J. Besl and R.C. Jain, Segmentation through variable order surface fitting, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(2) (1988) 167–192.

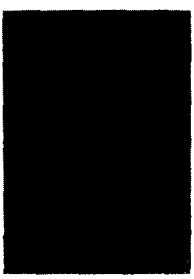
[4] G. Bilbro, M. White and W. Snyder, Image segmentation with neurocomputers, in: R. Eckmiller and C. von der Malsburg (eds.), *Neural Computers, NATO ASI Series: Vol. F41*, (Springer-Verlag, Berlin, Germany, 1987) 71–79.

[5] J.F. Canny, A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(6) (1986) 679-698.

[6] D. DeSieno, Adding a conscience to competitive learning, *Proc. of IEEE the Second International Conference on Neural Networks (ICNN88)* 1 (1988) 117–124.

[7] C.R. Dyer, Multiscale image understanding, in: L Uhr (ed.), *Parallel Computer Vision* (Academic Press, New York, NY, 1987) 171–213.

[8] K.S. Fu and J.K. Mui, A survey on image segmentation, *Pattern Recognition* 13 (1981) 3–16.
[9] R.M. Haralick and L.G. Shapiro, Survey, image segmentation techniques, *Computer Vision, Graphics Image Process* 29 (1985) 100–132.
[10] W.D. Hillis, *The Connection Machine* (MIT Press, Cambridge, MA, 1985).
[11] R. Hoffman and A.K. Jain, Segmentation and classification of range images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9(5) (1987) 608–620.
[12] A. Hurlbert and T. Poggio, A network for image segmentation using color, in: D.S. Touretzky (ed.), *Advances in Neural Information Processing Systems* (Morgan Kaufmann Publishers, San Mateo, CA, 1989) 297–304.
[13] J.M. Jolion and A. Rosenfeld, *A Pyramid Framework for Early Vision* (Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994).
[14] J. Koh, M. Suk and S.M. Bhandarkar, A multi-layer self-organizing feature map for range image segmentation, *Neural Networks* 8(1) (1995) 67–86.
[15] T. Kohonen, *Self-Organization and Associative Memory*, 2nd Edition (Springer-Verlag, Berlin, Germany, 1988).
[16] A. Leonardis, A. Gupta and R.M. Gray, Segmentation of range images as the search for geometric parametric models, *Proc. Intl. Conf. Computer Vision* (1990) 121–125.
[17] W. Lin, E. Tsao and C. Chen, Constraint satisfaction neural networks for image segmentation, in: T. Kohonen, K. Mkisara, O. Simula and J. Kangas (eds.), *Artificial Neural Networks* (Elsevier Science Publishers, 1991) 1087–1090.
[18] Y. Linde, A. Buzo and R.M. Gray, An algorithm for vector quantizer design, *IEEE Transactions on Communications* 28(1) (1980) 84–95.
[19] Y. Lu and R.C. Jain, Reasoning about edges in scale space, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(4) (1992) 450–468.
[20] D. Marr and H.K. Nishihara, Visual information processing: Artificial intelligence and the sensorium of sight, *Technology Review* 81(1) (1978) 2–23.
[21] G. Martinelli, L.P. Licotti and S. Ragazzini, Nonstationary lattice quantization by a self-organizing neural network, *Neural Networks* 3(4) (1990) 385–393.
[22] J. Naylor and K.P. Li, Analysis of a neural network algorithm for vector quantization of speech parameters, *Proc. of the 1st Annual INNS Meeting* (1988) 310–315.
[23] N.R. Pal and S.K. Pal, A review on image segmentation techniques, *Pattern Recognition* 26(9) (1993) 1277–1294.
[24] T. Pavlidis, *Algorithms for Graphics and Image Processing* (Computer Science Press, Rockville, MD, 1982).
[25] H. Ritter and K. Schulten, Kohonen's self-organizing maps: Exploring their computational capabilities, *Proc. IEEE Intl. Joint Conf. on Neural Networks (IJCNN88)* 1 (1988) 109–116.
[26] P.K. Sahoo, S. Soltani, A.K.C. Wong and Y.C. Chen, A survey of thresholding techniques, *Computer Vision, Graphics Image Process.* 41 (1988) 233–260.
[27] H. Samet, *The Design and Analysis of Spatial Data Structures* (Addison-Wesley Pub. Co., Reading, MA, 1990).
[28] A. Scherf and G. Roberts, Segmentation using neural networks for automatic thresholding, in: S. Rogers (ed.), *Proc. SPIE Conference on Applications of Artificial Neural Networks* (Orlando, FL, 1294, 1990) 118–124.
[29] A.P. Witkin, Scale space filtering, *Proceedings of the 8th International Conference on Artificial Intelligence* (1983) 1019–1021.
[30] S.M. Bhandarkar, J. Koh and M. Suk, A hierarchical neural network and its application to image segmentation, invited paper in *Int. J. Math. Comput. in Simulation* 41 (3–4) (1996) 337–355.



Suchendra M. Bhandarkar received his B.Tech in Electrical Engineering from Indian Institute of Technology, Bombay, India, in 1983, M.S. and Ph.D. in Computer Engineering from Syracuse University, Syracuse, New York in 1985 and 1989, respectively. He is currently an Associate Professor in the Department of Computer Science at the University of Georgia, Athens, Georgia. He was a Syracuse University Fellow for the academic years 1986–1987 and 1987–1988. He is a member of the IEEE, AAAI, ACM and SPIE and the honor societies Phi Kappa Phi and Phi Beta Delta. He is a co-author of the book *Object Recognition from Range Images* (Springer- Verlag, 1992). His research interests include computer vision, pattern recognition, image processing, artificial intelligence and parallel algorithms and architectures for computer vision and pattern recognition.

Jean Koh received his B.S. in Electrical Engineering from Seoul National University, Seoul, Korea in 1986, M.S. and Ph.D. degrees, in Computer Engineering in 1989 and 1994, respectively, from Syracuse University, Syracuse, New York. He is currently employed by Goldstar Inc., Seoul, Korea. His research interests are in the areas of computer vision, neural networks and parallel computing.

Minsoo Suk received his B.S., M.S. and Ph.D. degrees, all in Electrical Engineering in 1970, 1972 and 1974 respectively, from the University of California at Davis. He is an Associate Professor in the Department of Electrical and Computer Engineering at Syracuse University, Syracuse, New York. Prior to joining Syracuse University, he was a member of the Technical Staff at Rockwell International, an Associate Professor of Electrical Engineering at the Korea Advanced Institute of Science and Technology and a visiting Associate Professor at the University of California at Davis and the University of California at Santa Barbara. His current research interests are in the areas of image processing, computer vision, pattern recognition, neural networks and parallel computing.