Supervised Textured Image Segmentation Using Feature Smoothing and Probabilistic Relaxation Techniques

JOHN Y. HSIAO, MEMBER, IEEE, AND ALEXANDER A. SAWCHUK, SENIOR MEMBER, IEEE

Abstract—**This paper describes a supervised textured image segmentation algorithm. The goals are to improve textured image segmentation results, especially along the borders of regions; and to take into account the spatial relationship among pixels to improve the segmentation of region interiors.**

An improved method to extract textured energy features in the feature extraction stage is described. The method is based upon an adaptive noise smoothing concept which takes the nonstationary nature of the problem into account. Texture energy features are first estimated using a window of small size to reduce the possibility of mixing statistics along region borders. The estimated texture energy feature values are then smoothed by a quadrant filtering method to reduce the variability of the estimates while retaining the region border accuracy.

In this supervised segmentation system, the estimated feature values of each pixel are used by a Bayes classifier to make an initial probabilistic labeling. The spatial constraints are then enforced through the use of a probabilistic relaxation algorithm. Two probabilistic relaxation algorithms are investigated and their results are compared.

Limiting the probability labels by probability threshold is proposed to make the relaxation iteration more efficient. The tradeoff between efficiency and degradation of performance is studied. Finally, an overview of the proposed textured image segmentation system is provided and comparisons of overall performance are made.

Index Terms—**Relaxation, segmentation, smoothing, supervised classification, texture analysis, texture features.**

I. Introduction

An important task of an image analysis system is to segment the given image into meaningful regions and to label the individual regions. Segmentation of the image into regions can be done in many ways [1], [2]: from simple pixel classification methods using gray levels and local feature values to sophisticated split-and-merge techniques using statistical homogeneity tests. These segmented and labeled regions can then be used for representation, description, and recognition.

There exist two major approaches to image segmentation: edge-based and region-based. In edge-based methods, the local discontinuities are detected first and then are connected to form longer boundaries. In region-based methods, areas of the image with homogeneous properties are found, which in turn give the boundaries. The presence of texture causes difficulty for both the edge- and region-based methods. Essentially, our hypothesis that the objects consist of relatively homogeneous surfaces is violated, and very few intensity changes now correspond to object boundaries. In edge-based methods, we will get many edges due to texture that must be differentiated from object boundaries; in region-based methods, we will get many small regions that correspond to texture. Therefore, textured image segmentation is an important and difficult task.

What should a good image segmentation be? As Haralick [3] pointed out, a good image segmentation should possess the following properties. Regions of an image segmentation should be uniform and homogeneous with respect to some characteristic such as gray level or texture; adjacent regions of segmentation should have significantly different values with respect to the characteristic on which they are uniform; region interiors should be simple and without many small holes; and boundaries of each segment should be simple, not ragged, and must be spatially accurate.

In order to achieve the objective of designing effective algorithms which could provide us with the properties pointed out by Haralick, we think the following ingredients are essential for a textured image segmentation system [4], [5], [6]: a set of texture features having good discriminating power; a segmentation algorithm having spatial constraints; and estimation of texture features taking the nonstationary nature of the feature image planes into account.

Since developing texture features is not our main concern, we decided to choose a set of existing texture features which can provide us good discriminating power and are easy to compute to serve our need. After a careful study, we found Laws' texture energy feature set [7], [8] meets our criteria. Laws' set of features has been shown to work as well or better than most others in texture classification problems. However, when Laws applied the texture energy features to the segmentation problem, only limited success was achieved. There are considerable er-

Manuscript received October 28, 1987; revised February 13, 1988. Recommended for acceptance by C. Brown. This work was supported in part by the Joint Services Electronics Program (JSEP) through the Air Force Office of Scientific Research under Contract F49620-85-C-0071. J. Y. Hsiao was supported by the Hughes Doctoral Fellowship Program.

J. Y. Hsiao is with Hughes Aircraft Company, Electro-Optical and Data Systems Group, Los Angeles, CA 90245.

A. A. Sawchuk is with the Department of Electrical Engineering—Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089.

IEEE Log Number 8929339.

rors in the interiors of the large regions and the algorithm sometimes performs badly near the borders between the textures. In this paper we identify two areas in the Laws segmentor that still need improvement, and suggest solutions to them.

First, we describe an improved method to extract texture energy features for segmentation. We hope to improve segmentation results, especially along the borders of regions. One important point is that we think more emphasis should be put into the early stages of the segmentation process (such as feature extraction) in order to improve the overall performance.

Second, we think the incorporation of spatial constraints into the segmentation algorithm is essential. We suggest the use of the probabilistic relaxation method as a means to enforce the spatial constraints into our segmentation algorithm.

There has been numerous previous work on the textured image segmentation problem. Basically, textured image segmentation approaches are characterized by two features: the texture measure and the segmentation algorithm. Commonly used texture measures include: Fourier transform domain texture energy [9], [10]; number of local extrema [11]; co-occurrence matrix [12]; mean and covariance [13]; spatial texture energy [7]; local orientation and frequency [14]; coarseness [15]; second-order gray level statistic [16]; autoregressive moving average (ARMA) model [17]; Gaussian-Markov random field (GMRF) model [18]; and Gibbs random field model [19]. Segmentation algorithms used by various researchers include: region growing [9], [10], [11], [13], [20]; clustering and thresholding [21], [22], [23], [15]; and estimation theoretic approaches [24], [17], [25], [18], [19].

II. An Improved Method to Extract Texture Energy Features

A. *Review of Laws Textured Image Segmentation System*

Laws [7], [8] proposed an approach to measure texture energy in the spatial domain. His approach to texture characterization consists of two steps. First the image is convolved with a set of filters having a small region of support in the spatial domain. These filters are called microtexture masks, and their outputs are called microtexture features. These masks contain weighting coefficients needed in the two-dimensional convolution process. The two-dimensional convolution of the image $f(i, j)$ and mask $h(i, j)$ with size $2a + 1$ by $2a + 1$ is given by the relation

$$x(i,j) = (h * f)(i,j) = \sum_{k=-a}^{a} \sum_{l=-a}^{a} h(k, l) f(i + k, j + l), \quad (1)$$

for $i = 0, 1, \cdots, N - 1$ and $j = 0, 1, \cdots, N - 1$ where $*$ denotes two-dimensional convolution. Laws' microtexture masks are designed to act as matched filters for certain types of quasiperiodic variations commonly found in textured images. Typically, these masks are of size 7×7 or smaller. In most cases, the sum of the elements of the mask is zero, which results in the output image having a mean of zero. The convolution masks are intended to be sensitive to visual structure such as edges, ripples, and spots. Fig. 1 shows the coefficients of four microtexture masks that had the best discrimination power for the texture mosaic that Laws used. Laws' texture mosaic is similar to ours, which is shown in Fig. 5(a). In this paper, we use only the four microtexture features derived from the masks in Fig. 1. Our goal here is the improvement of segmentation at borders using smoothing and relaxation techniques applied to a given feature set, rather than the derivation of an optimum feature set for a given set or class of textures. Because the microtexture features are quasi-periodic, we expect strong variations about the mean output as a function of mask position for masks that are matched to the local texture. Thus, the relevant information for texture discrimination is now present as the image variance of the microtexture feature planes.

$$E5L5 = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad R5R5 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

$$E5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix} \quad L5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

Fig. 1. Four of Laws' microtexture masks.

Second, macrostatistical features are obtained over large windows (e.g., 15×15 or 31×31). The most useful statistics are the sample variances or a computationally more efficient statistic, called ABSAVE (or sample mean deviation), of the microtexture feature planes. The ABSAVE $s(i, j)$ is defined as the mean deviation within a $2n + 1$ by $2n + 1$ window at point (i, j) and is given by

$$s(i, j) = \frac{1}{(2n + 1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} \left| x(k, l) - m(i, j) \right| \quad (2)$$

where the mean $m(i, j)$ is given by

$$m(i, j) = \frac{1}{(2n + 1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} x(k, l). \quad (3)$$

Based upon the assumption that the mean is zero, the ABSAVE $s(i, j)$ becomes

$$s(i, j) = \frac{1}{(2n + 1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} \left| x(k, l) \right|. \quad (4)$$

These features, i.e., the ABSAVE of each microtexture filtered image, were called texture energy measures by Laws.

Once these texture energy features are extracted, an optional feature selection step was used to reduce the dimensionality of the classification process. There are two kinds of feature selection and extraction methods depending on whether the labeled prototypes for classes and their statistics are available or not. If the prototype labels are known, the multiple discriminant analysis method [26], a generalization of Fisher's linear discriminant to the multiple class problem, was used to reduce the dimensionality of the feature space. Then, a minimum distance-to-centroids classifier was used to assign the label of test observation. No spatial information was included in this procedure.

If labeled prototypes were not available, Laws used a different feature extraction method called principal components analysis [26]. The absence of class membership information of the training observations implies that the class-conditional parameters cannot be inferred from the data. In such a situation the only sensible approach to feature extraction is to take advantage of the general information-compression properties of the Karhunen-Loéve (K-L) expansion. After the principal component planes were generated, the first three were then input to the "Ohlander segmentor" algorithm [27]. The Ohlander segmentor is a region based segmentor. Regions are split recursively based upon histogram analysis of the feature values. It begins by defining a mask that initially covers all pixels of the image. Given a mask, a histogram of the masked image is computed. The separation of one mode of the histogram in the feature space from another mode is then followed. Pixels on the image are then identified with the cluster to which they belong. If there is only one feature space cluster, then the mask is terminated. If there is more than one cluster, then each spatially connected component of all pixels with the same cluster is, in turn, used to generate a mask which is placed on a mask stack. During successive iterations the next mask in the stack selects pixels in the histogram computation process. Clustering is repeated for each new mask until the stack is empty [3].

B. Potential Problems

As we mentioned before, when Laws applied texture energy features to the segmentation problem, there were considerable errors present in the interiors of the large regions and near the borders between the textures. One major reason for the segmentation error along the region borders is due to the method of estimating the local texture energy. As we recall, the macrostatistical features are obtained by processing the microtexture feature planes with a nonlinear "local texture energy" filter. This nonlinear filter was used to estimate the local sample standard deviation (or a similar statistic approximated by the moving window average of the absolute values) of the filtered image. Although such moving window operations are simple and fast, it introduces significant errors along the region borders. The reason for this is that the window used to calculate macrostatistical feature often overlaps more than one texture at region boundaries. When this occurs, the resulting statistics become the mixture of two sets of statistics. This mixture of statistics sometimes resemble another distinctly different texture class. In this case, the boundary pixels will be incorrectly classified as a third class different from the two inside the overlapping windows.

One way to avoid the aforementioned problem is to estimate the sample standard deviation (or sample mean deviation) of the microtexture filtered images more carefully. We can perceive the process of getting the macrostatistical features as the problem of smoothing a noisy image. In noise smoothing we always face the following problem: how to smooth noise without blurring the edges. There are many ways to approach the noise smoothing problem, and we now extend some of these concepts to the feature extraction problem.

C. Edge Preserving Noise Smoothing Methods

The locally linear minimum mean square error (LLMMSE) estimator suggested by Kuan *et al.* [28] has the property that it smoothes out noise in flat regions and leaves the observation unchanged in the vicinity of edges. The performance of the LLMMSE filter strongly depends on how the local windows are chosen and how the local statistics are calculated. Jiang-Sawchuk [29] suggested a quadrant method to estimate the local statistics. The local variance $v_g(i, j)$ for the noisy observation g at (i, j) was computed in the quadrant method by

$$v_g(i, j) = \min\left[v_{g1}(i, j), v_{g2}(i, j), v_{g3}(i, j), v_{g4}(i, j)\right] \tag{5}$$

where $v_{g1}(i, j)$, $v_{g2}(i, j)$, $v_{g3}(i, j)$, and $v_{g4}(i, j)$ are the four quadrant variances defined by

$$v_{g1}(i, j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i + k, j - l) - m_g(i + k, j - l)\right]^2, \tag{6}$$

$$v_{g2}(i, j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i - k, j - l) - m_g(i - k, j - l)\right]^2, \tag{7}$$

$$v_{g3}(i, j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i - k, j + l) - m_g(i - k, j + l)\right]^2, \tag{8}$$

$$v_{g4}(i, j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i + k, j + l) - m_g(i + k, j + l)\right]^2 \tag{9}$$

where the local mean is

$$m_g(i, j) = \frac{1}{(2m + 1)^2} \sum_{k=-m}^{+m} \sum_{l=-m}^{+m} g(i + k, j + l). \tag{10}$$

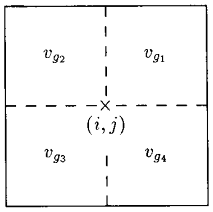


Fig. 2. Location of the four quadrant variances.

The locations of the neighboring pixels involved in the above four quadrant variances are schematically shown in Fig. 2. Note that the window sizes for calculating local mean and quadrant variances are not necessarily the same.

A more sophisticated edge preserving smoothing algorithm was proposed by Nagao and Matsuyama [30]. Nine windows, each containing a few pixels, were used to estimate local statistics at each pixel. The most homogeneous neighborhood area among these nine regions was then selected. The averaged gray level of the selected neighborhood area is used as the estimate for the pixel under consideration. Noise in an image is removed by the repeated usage of this method, while the edges remain sharp. For complex-shaped regions, this method gives better result than quadrant window does, at the expense of higher computing load.

For large and simple-shaped regions, which is the assumption in this work, using four quadrant windows as shown in Fig. 2 to estimate local statistics is adequate. In summary, in our method quandrant windows around each pixel are used to estimate local statistics and then the averaged gray level of the most homogeneous neighborhood as determined by (5) is used to replace the pixel under consideration. If a window contains a sharp edge, the variance of the gray level in that window becomes large. Here the quadrant variance as defined by (5) is used to measure the gray level variability around a pixel to find the most homogeneous neighborhood. For convenience, we call this method the Edge Preserving Noise Smoothing Quadrant (EPNSQ) filter.

1) Simulation Results: Fig. 3 shows the simulation results of using a simple averaging filter and LLMMSE filter to smooth a noisy image. Fig. 3(a) is the original image. Fig. 3(b) is the degraded image with a signal-independent additive white noise. The signal-to-noise ratio (SNR) of the degraded image is about 6 dB. We define the SNR as the ratio of the signal variance to the noise variance. The signal variance is defined as the signal sample variance of the whole image. Fig. 3(c) is the simple averaging filter output which is the degraded image convolved with a 7×7 uniform window. The result of using the LLMMSE and quadrant method with $m = 3$ in (10) and $w = 4$ in (6)–(9) for variance estimation is shown in Fig. 3(d). The sharpness of edges is retained while the noise is smoothed out as compared to the original.

The simulation results of using the EPNSQ filter with quadrant windows of size 7×7 is shown in Fig. 4. As before, the sharpness of edges and the smoothness of interior regions gives a subjectively better result.

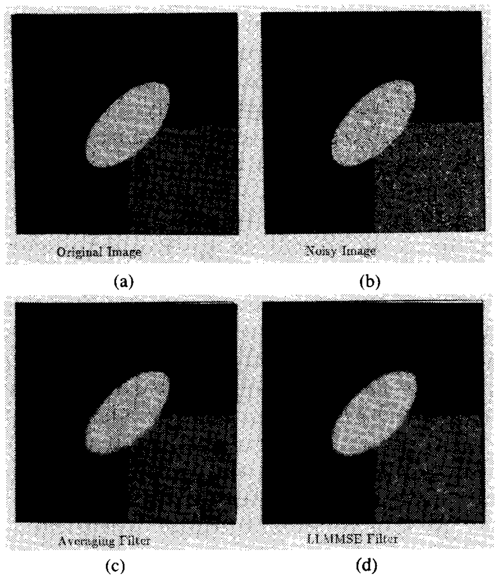


Fig. 3. Comparison between an averaging filter and the LLMMSE filter.

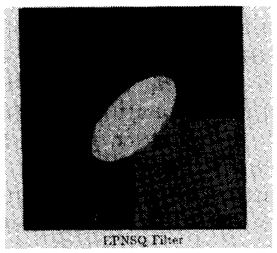


Fig. 4. The EPNSQ filtered result.

D. Using the EPNSQ Filter to Smooth Feature Estimates

Our problem on hand can be formulated as how to estimate "texture energy" features from the microtexture filtered images with less possibility of mixture of statistics along region boundaries. This is very similar to the problem of smoothing noise without blurring edges.

We propose a two stage scheme with varying local window sizes to estimate and smooth "texture energy" features (see Fig. 6). Each of the microtexture filtered output images $x(i, j)$ is first processed by a 7×7 "local texture energy" filter to generate an image $s(i, j)$. In other words, each pixel of $s(i, j)$ is an estimate of the local sample standard deviation for a particular microtexture filter. By using such a small window to estimate the local standard deviation, the variability of the estimate is still quite high. In order to reduce the variability of the estimates yet not sacrifice the region border accuracy, we now adapt the EPNSQ filter used for gray level image smoothing to feature smoothing. Instead of indiscriminately smoothing at each point, the EPNSQ filter smoothes $s(i, j)$ at chosen neighbors that are likely to belong to the same region as the given point. Because our main concern is to avoid averaging over region borders, the problem becomes one of finding a measure which indicates whether there is a region border or not.

As previously discussed, we know the sample variance can be used as a measure of the nonhomogeneity of regions. This idea can be extended to the feature smoothing problem. Now in our EPNSQ filter we use the sample variance of "texture energy feature" as the measure for choosing the most homogeneous neighborhood. At each pixel a set of four neighborhoods that lie on various sides of the point are examined. The sample mean m_{W_1} and sample variance v_{W_1} of W_1 defined by

$$m_{W_1}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i+k, j-l), \tag{11}$$

and

$$v_{W_1}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[s(i+k, j-l) - m_{W_1}(i,j)\right]^2, \tag{12}$$

are computed where $s(i, j)$ represents the local sample deviation at location (i, j). Similarly, the sample means and sample variances of windows W_2, W_3, and W_4 are defined as

$$m_{W_2}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i-k, j-l), \tag{13}$$

$$m_{W_3}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i-k, j+l), \tag{14}$$

$$m_{W_4}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i+k, j+l), \tag{15}$$

$$v_{W_2}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[s(i-k, j-l) - m_{W_2}(i,j)\right]^2, \tag{16}$$

$$v_{W_3}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[s(i-k, j+l) - m_{W_3}(i,j)\right]^2, \tag{17}$$

$$v_{W_4}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[s(i+k, j+l) - m_{W_4}(i,j)\right]^2. \tag{18}$$

Those windows which contain region borders generally have a higher variability introduced by the edges.

1) Window Sizes: Another issue that needs to be addressed is the effect of using different window sizes to estimate the local sample statistics. We assume the class of textures we are working with are those fine-grained textures typically used in our experiment. This assumption is made because statistical methods are usually suitable for microtextures, i.e., textures with short correlation distance. Otherwise, methods such as structural or structural-statistical [31] may be a better choice.

The choice of window sizes used to estimate the local statistics is actually a tradeoff problem. Using small window sizes (e.g., 7×7), we can estimate the statistics more accurately near the borders and in the small regions, but we have higher statistical variability compared to the use of a large window due to the smaller number of data points. On the other hand, using large window sizes (e.g., 15×15) provides more data points to reduce the statistical variability of the estimates in the large regions, but performs poorly near the borders between the textures and in the small regions. In our two stage scheme, a small window size (e.g., 7×7) is first used to estimate the local statistics. The choice of this macrostatistical window size was based upon Laws' study which indicated that with a 7×7 window a reasonably good classification rate can still be achieved. A larger window size (e.g., 15×15) EPNSQ filter is then used to provide more smoothing power. Since region borders have already been accurately estimated by the smaller macrostatistical window, the EPNSQ filter should be able to identify and smooth those homogeneous regions. The window size used by the EPNSQ filter should be smaller than the smallest region size which we expect to segment accurately.

III. Supervised Classification Using a Bayes Classifier

A. Algorithm Description

The block diagram of our algorithm is depicted in Fig. 6. The following is a description of the algorithm.

1) The textured image shown in Fig. 5 is first convolved with a set of 5×5 microtexture masks. In Laws' work, a principal component transformation was then used to allow selection of a working subset containing the most significant transformed features. In our work, because feature selection is not our main concern, the output of a subset of four microtexture masks is used without performing any principal component rotation or feature selection. These masks, which correspond to the E5L5, E5S5, L5S5 and R5R5 masks (shown in Fig. 1), were chosen because they were indicated by Laws to be the most important for classifying the textures used by him which are similar to ours. Different sets of masks or a different number of masks could be used in other applications.

2) Because the microtexture masks are zero-sum, the resulting filtered images have mean that are close to zero, and the local standard deviation may be approximated by averaging the absolute values within the window. In our work, we used a 7×7 window to estimate this statistic.

3) The local standard deviation estimated by the 7×7 window in step 2) is now smoothed by the EPNSQ filter. At each pixel, we examine a set of four neighborhoods that lie on various sides of the pixel, then use local sample variance as the measure of variability over each neigh-

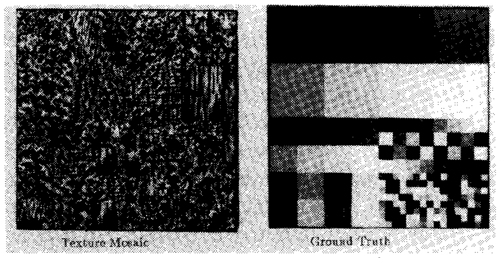


Fig. 5. Texture mosaic 1 for experimental work.

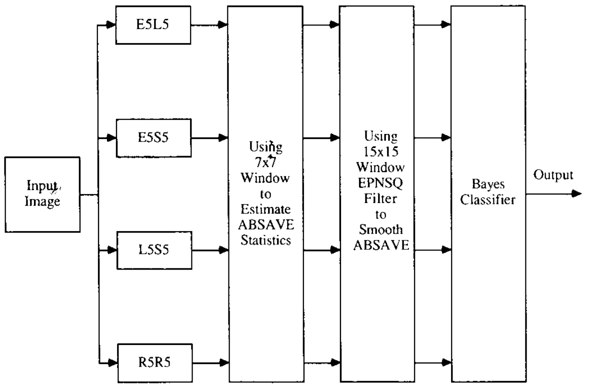

E5L5
E5S5
L5S5
R5R5
Input Image
Using 7x7 Window to Estimate ABSAVE Statistics
Using 15x15 Window EPNSQ Filter to Smooth ABSAVE
Bayes Classifier
Output

Fig. 6. Block diagram of our segmentor with Bayes classifier.

borhood, and replace the pixel by the average of the neighborhood that has the lowest variance. This selects the neighborhood that is most likely to lie entirely within a uniform region of the picture. This EPNSQ filtering was applied to each of the four images representing the feature plane outputs of the four original microtexture masks. In this step we use a window size 15 × 15.

4) Each of the estimated feature vectors $\vec{x}$ is then sent to a Bayes classifier. If we assume the class conditional probability density for class λ_i is multivariate Gaussian with mean $\vec{m}_i$ and covariance matrices Σ_{λ_i}, and the *a priori* probabilities are equal for all classes, then classification can be achieved by use of the discriminant functions [32]

$$g_i(\vec{x}) = (\vec{x} - \vec{m}_i)^t \Sigma_{\lambda_i}^{-1}(\vec{x} - \vec{m}_i) + \log |\Sigma_{\lambda_i}|. \quad (19)$$

The Bayes decision rule classifies a point to class λ_i if and only if

$$g_i(\vec{x}) < g_j(\vec{x}) \qquad \text{for all } j \neq i. \quad (20)$$

The output of the Bayes classifier is a gray level coded classification result.

B. Simulation Results

The first test image used in our work is a texture mosaic [Fig. 5(a)] which consists of eight different textures: grass, water, sand, wool, pigskin, leather, raffia, and wood. The texture images we have chosen are from an album by Brodatz [33]. All eight textures are present in the image in squares of size 128 × 128, 64 × 64, 32 × 32, and 16 × 16. All components are histogram equalized so that all textures have same first order statistics. In other words, first-order statistics alone are not sufficient for discriminating different textures. Fig. 5(b) is the gray level coded ground truth of Fig. 5(a).

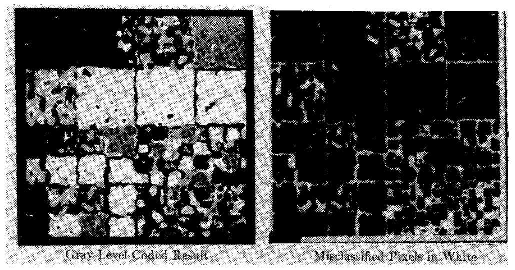


Fig. 7. Classification results using 15 × 15 macrostatistical window.

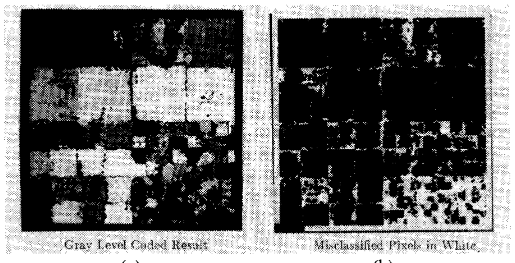


Fig. 8. Classification results using our method.

Fig. 7(a) shows the gray level coded classification result using feature data estimated by Laws' method. The window size used for estimating statistics is 15 × 15. Note the accuracy near some of the borders between textures is not very satisfactory. Fig. 7(b) is a binary image of the classification result in which the black pixels represent correctly classified ones, the white pixels represent misclassified ones.

Fig. 8(a) is the gray level coded classification result using our algorithm to estimate the feature data. A 7 × 7 window is used for statistics estimation and a 15 × 15 window is used for EPNSQ smoothing. Fig. 8(b) shows the binary image of the classification result in which the black pixels represent correctly classified ones, the white pixels represent misclassified ones. Table I lists the correct classification rate of the overall image and of each subimage containing regions of a particular size. From Fig. 8 and Table I we can see not only the correct classification rate but also the accuracy along borders has been improved (except in the region containing 16 × 16 sizes, an area which is very difficult for a human to discriminate).

For our test image, the chosen window sizes, i.e., 5 × 5 for microtexture masks, 7 × 7 for macrostatistics estimation, and 15 × 15 for EPNSQ filter, worked very well. This sequence of window sizes may not be a good choice for all images. In other words, to choose a appropriate sequence of window sizes is a data dependent problem. In the process of choosing window sizes, the following guidelines must be considered.

TABLE I
CLASSIFICATION ACCURACY (%) OF TWO FEATURE EXTRACTION SCHEMES

Region Sizes	15 x 15	7 x 7-15 x 15 EPNSQ Filter
Overall	72.9	76.6
128 x 128	81.1	88.1
64 x 64	74.1	78.6
32 x 32	64.5	66.1
16 x 16	44.3	37.3

- The window sizes for microtexture masks are typically 3×3, 5×5, or 7×7, depending on the spatial scale size of the microtexture features.
- The window size used for macro statistic estimation should be large enough to include a representative sample of the image texture. The coarser the texture the larger the window size should be.
- The window size used for EPNSQ filter should be smaller than the smallest region size in which we expect to have a good segmentation.

IV. Supervised Segmentation Algorithm with Spatial Constraints

A. Introduction

The supervised segmentation technique described in Section III has primarily ignored the spatial relationship between pixels. The weakness of classifying pixels based solely upon feature space distribution is that the formation of clusters in the feature space does not take into consideration the spatial distribution of points in the image. In addition, the mapping of a single class label back to the image is only a gross representation of feature space information, which loses the relationship of each point to other classes in feature space. These observations suggest that if we want to obtain better segmentation performance, we must make use not only of similarities among the pixels, but also of their relative positions.

In this section, we use the probabilistic relaxation method as a means to introduce spatial constraints into our segmentation algorithm. Other methods, such as the one suggested by Faugeras and Berthod [34], may be used to bring in contextural information. Section IV-B reviews some of the past work that uses relaxation in classifying pixels. Section IV-C discusses how we use the probabilistic relaxation technique in our textured image segmentation problem. Section IV-D presents the supervised segmentation simulation results.

B. Review of the Probabilistic Relaxation Method in Pixel Classification

The idea of effective use of cooperative processing through the mechanism of probabilistic relaxation was first introduced by Rosenfeld *et al.* [35]. Probabilistic relaxation is an iterative approach for using contextural information to reduce local ambiguities. Let us first review some of the concepts involved in probabilistic relaxation as well as some past work that uses relaxation in classifying pixels.

The probabilistic relaxation process involves a set of objects $A = \{a_1, a_2, \cdots, a_N\}$ and a set of class labels $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_M\}$. In pixel classification problems, the set of objects in general is the whole set of pixels in the image represented in lexicograpic notation. For each object a_i we are given a set of local measurements, which are used as a basis for estimating the probabilities $P_i(\lambda_k)$ of object a_i having label λ_k, $k = 1, \cdots, M$. These probabilities satisfy the condition

$$\sum_{k=1}^{M} P_i(\lambda_k) = 1, \qquad \text{for all} \quad a_i \in A,$$
$$\text{and} \quad 0 \leq P_i(\lambda_k) \leq 1. \tag{21}$$

Suppose that the class label assignments of the objects are interdependent; in other words, for each pair of class label assignments $a_i \in \lambda_k$ and $a_j \in \lambda_l$, we have some quantitative measure of the compatibility of this pair, denoted by $r_{ij}(\lambda_k, \lambda_l)$. There are many ways to choose and to interpret the compatibility coefficients. Peleg and Rosenfeld [36] suggested that compatibility coefficients may be either interpreted as correlations or mutual information. Zucker and Mohammed [37] suggested rewriting and modifying the relaxation in such a way that the compatibility coefficients had the meaning of conditional probabilities. Nagin *et al.* [38] suggested using orientation dependent compatibility coefficients in their global and local histogram-guided relaxation algorithms.

There are also many possible iteration schemes that can be used to update the labeling probabilities based upon initial probabilities and compatibility coefficients. We discuss two of such iterative updating schemes here: the original relaxation iteration scheme developed by Rosenfeld *et al.* [35] and an improved probabilistic relaxation scheme developed by Peleg [39].

Rosenfeld *et al.* [35] have emphasized the importance of graph labeling and proposed an approximate, iterative, parallel relaxation method as a solution to it. The probabilistic relaxation approach was then used by Eklundh *et al.* [40] in a multispectral pixel classification problem. Their experiment was conducted on a color picture of house. The picture was hand segmented into five regions, which were regarded as the ground truth when evaluating results. The means and covariance matrices for the five classes were then estimated. Initial estimates of the probabilities of membership in these classes were made by assuming the clusters to be normally distributed, and using the hand segmentation to define the prior probability $P(\lambda_k)$ for class λ_k. Because they assumed the class conditional density function for class λ_k is a d-dimensional multivariate Gaussian density, i.e.,

$$p(\vec{x} \mid \lambda_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_{\lambda_k}|^{1/2}} \cdot \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_{\lambda_k})^t \Sigma_{\lambda_k}^{-1} (\vec{x} - \vec{\mu}_{\lambda_k})\right). \tag{22}$$

Pixel a_i is assigned to class λ_k with probability:

$$P_i(\lambda_k | \vec{x}) = \frac{p(\vec{x} | \lambda_k) P(\lambda_k)}{\sum_{j=1}^{M} p(\vec{x} | \lambda_j) P(\lambda_j)} \qquad k = 1, \cdots, M. \tag{23}$$

These probabilities were used as the initial probabilities $P_i^{(0)}(\lambda_k) = P_i(\lambda_k | \vec{x})$.

They defined the compatibility coefficients based on the mutual information of the labels at neighboring points. The probability of any point having the label λ_k may be estimated by

$$\hat{P}(\lambda_k) = \frac{1}{N} \sum_{i=1}^{N} P_i^{(0)}(\lambda_k), \tag{24}$$

and the joint probability of a pair of points having labels λ_k and λ_j by

$$\hat{P}_{i,i+\delta}(\lambda_k, \lambda_j) = \frac{1}{N} \sum_{i=1}^{N} P_i^{(0)}(\lambda_k) P_{i+\delta}^{(0)}(\lambda_j) \qquad \delta \in \Delta \tag{25}$$

where N is the number of points, pixel $i + \delta$ is a specific neighbor of pixel i, and Δ defines the neighborhood about the particular pixel being considered. An estimate of the mutual information that the contribution of λ_j to the information about λ_k is expressed as

$$I_{i,i+\delta}(\lambda_k, \lambda_j) = \ln \frac{\hat{P}_{i,i+\delta}(\lambda_k, \lambda_j)}{\hat{P}(\lambda_k) \hat{P}(\lambda_j)} \qquad \delta \in \Delta. \tag{26}$$

Assume the events that cause

$$\frac{P_{i,i+\delta}(\lambda_k, \lambda_j)}{\hat{P}(\lambda_k) \hat{P}(\lambda_j)} \qquad \delta \in \Delta, \tag{27}$$

to be outside the range $[e^{-5}, e^{5}]$ can be ignored. Thus values of $I_{i,i+\delta}(\lambda_k, \lambda_j)$ can be considered to lie in the range $[-5, 5]$ and the compatibility coefficient is defined as

$$r_{i,i+\delta}(\lambda_k, \lambda_j) = \tfrac{1}{5} I_{i,i+\delta}(\lambda_k, \lambda_j) \qquad \delta \in \Delta, \tag{28}$$

which lie in the range $[-1, 1]$. The compatibility coefficients are fixed throughout the process.

The compatibility coefficients can then be used to compute the updating factor

$$q_i^{(m)}(\lambda_k) = \sum_{\delta \in \Delta} d_{i+\delta} \sum_{j=1}^{M} r_{i,i+\delta}(\lambda_k, \lambda_j) P_{i+\delta}^{(m)}(\lambda_j) \tag{29}$$

where $d_{i+\delta}$, $\delta \in \Delta$, are a set of neighbor weights that can be used to give different neighbors differing degrees of influence in the neighborhood function. Eklundh *et al.* chose the set of neighbors as the 8-connected neighbors plus the central pixel (i.e., that under consideration). The relaxation iterations have the form

$$P_i^{(m+1)}(\lambda_k) = \frac{P_i^{(m)}(\lambda_k)[1 + q_i^{(m)}(\lambda_k)]}{\sum_{j=1}^{M} P_i^{(m)}(\lambda_j)[1 + q_i^{(m)}(\lambda_j)]} \qquad k = 1, \cdots, M. \tag{30}$$

In their experiment they iterated until the classification error was a minimum.

Peleg [39] suggested that the compatibility coefficients $r_{i,i+\delta}(\lambda_k, \lambda_j)$ should take the form

$$r_{i,i+\delta}(\lambda_k, \lambda_j) = \frac{\hat{P}_{i,i+\delta}(\lambda_k, \lambda_j)}{\hat{P}(\lambda_k) \hat{P}(\lambda_j)} \qquad \delta \in \Delta \tag{31}$$

where $\hat{P}_{i,i+\delta}(\lambda_k, \lambda_j)$ and $\hat{P}(\lambda_k)$ [or $\hat{P}(\lambda_j)$] are defined in (25) and (24), respectively.

After finding the coefficients, the updating process can take place. For a specific $i + \delta$, we compute

$$S_{i,i+\delta}^{(m+1)}(\lambda_k) = \sum_{j=1}^{M} \hat{P}_i^{(m)}(\lambda_k) \hat{P}_{i+\delta}^{(m)}(\lambda_j) r_{i,i+\delta}(\lambda_k, \lambda_j) \qquad \delta \in \Delta, \tag{32}$$

and

$$q_{i,i+\delta}^{(m+1)}(\lambda_k) = \frac{S_{i,i+\delta}^{(m+1)}(\lambda_k)}{\sum_{j=1}^{M} S_{i,i+\delta}^{(m+1)}(\lambda_j)} \qquad \delta \in \Delta, \tag{33}$$

here $q_{i,i+\delta}^{(m+1)}(\lambda_k)$ is the new probability estimate for the labels at pixel a_i based upon the previous estimates at a_i and $a_{i+\delta}$. For the case of using eight connected neighbors, the new $P_i^{(m+1)}(\lambda_k)$ will be the average of these eight estimates:

$$P_i^{(m+1)}(\lambda_k) = \tfrac{1}{8} \sum_{\delta \in \Delta} q_{i,i+\delta}^{(m+1)}(\lambda_k) \qquad k = 1, \cdots, M. \tag{34}$$

The final pixel classification is the maximum-probability class for each pixel.

Some theoretical studies of the convergence properties of relaxation schemes have been conducted in Haralick *et al.* [41], Zucker *et al.* [37] and Hummel and Zucker [42], but we will not treat them here. There are several guidelines for evaluating the relaxation algorithms' performance [1].

- The sum of absolute probability differences $\Sigma_{i \in A} | P_i^{(m)}(\lambda_k) - P_i^{(m+1)}(\lambda_k) |$, for each $\lambda_k \in \Lambda$, should become small after a few iterations.
- The final probabilities should not be too far away, on the average, from the initial ones; we would not be satisfied with the process if it converged to an arbitrary set of final probabilities unrelated to the initial ones. Thus, $\Sigma_{i \in A} | P_i^{(m)}(\lambda_k) - P_i^{(0)}(\lambda_k) |$, for each $\lambda_k \in \Lambda$, should not become very large.
- The entropy of the probabilistic classification after applying relaxation should be less than the entropy of the

initial one. In other words, we expect, for each $\lambda_k \in \Lambda$,

$$-\sum_{i \in A} P_i^{(m)}(\lambda_k) \log P_i^{(m)}(\lambda_k) < -\sum_{i \in A} P_i^{(0)}(\lambda_k) \cdot \log P_i^{(0)}(\lambda_k).$$

In the next section, we will discuss how to apply the probabilistic relaxation method to our supervised textured image segmentation problem. To the best of the authors' knowledge, no previous attempts have been made in using probabilistic relaxation to solve the textured image segmentation problem.

C. Using Probabilistic Relaxation in Textured Image Segmentation

Assuming that a prototype of each texture under test is given, we use the method described in Section II to generate the texture energy features. We also assume that the class conditional density function for each of the texture classes is an d-dimensional multivariate Gaussian density as described in (22). In practice, the mean vector and covariance matrix for each class are unknown and must be estimated from prototypes. Let $\hat{\mu}_{\lambda_k}$ and $\hat{\Sigma}_{\lambda_k}$ be unbiased estimates of $\hat{\mu}_{\lambda_k}$ and Σ_{λ_k}. Then $\hat{\mu}_{\lambda_k}$ and Σ_{λ_k} are given by

$$\hat{\mu}_{kj} = \frac{1}{n_k} \sum_{l=1}^{n_k} x_{jl} \qquad j = 1, 2, \cdots, d, \tag{35}$$

$$\hat{\sigma}_{kjm} = \frac{1}{n_k - 1} \sum_{l=1}^{n_k} (x_{jl} - \hat{\mu}_{kj})(x_{ml} - \hat{\mu}_{km}) \qquad j = 1, 2, \cdots, d; \quad m = 1, 2, \cdots, d \tag{36}$$

where n_k is the number of prototype samples in class λ_k.

Assuming the *a priori* probability of each class $P(\lambda_k)$ is equal and the class conditional probability density function $p(\vec{x} \mid \lambda_k)$ is estimated as previously described, then the *a posteriori* probability of assigning a pixel with feature vector $\vec{x}$ to class λ_k is given by (23). These *a posteriori* probabilities of each pixel may be used as the initial probability labeling $P_i^{(0)}(\lambda_k)$ and compatibility coefficients may then be calculated.

Once the initial labeling probabilities of each pixel are generated, there are several ways to iteratively update them. Two algorithms, one was developed by Rosenfeld-Hummel-Zucker (RHZ) and the other was developed by Peleg, are chosen here for detailed study. In our study, we assume the set of neighbors of a pixel to be the 8-connected neighbors.

The RHZ probabilistic relaxation algorithm may be summarized as follows.

1) The mutual information coefficients are chosen to be the compatibility coefficients. For each one of the 8-connected neighbors, we use (24)–(28) to calculate the compatibility coefficients. The coefficients $r_{i,i+\delta}(\lambda_k, \lambda_j)$ are fixed throughout the iteration process.

2) With the initial labeling probabilities of each pixel and the compatibility coefficients on hand, we may start the iteration by first computing the updating factor as described in (29). Let Δ be the set of 8-connected neighbors of the pixel under consideration and $d_{i+\delta}$ equals $1/8$.

3) The updating factor is then used to compute $P_i^{(m+1)}(\lambda_k)$ according to (30).

4) Repeat steps 2)–3) until the stopping criteria is met. In our case, we specified the number of iterations in advance.

Similarly, the Peleg probabilistic relaxation algorithm is summarized as follows.

1) For each one of the 8-connected neighbors, we use (31) to calculate the compatibility coefficients. The coefficients $r_{i,i+\delta}(\lambda_k, \lambda_j)$ are fixed throughout the iteration process.

2) The iterative updating process follows by using (32)–(33). The $q_{i,i+\delta}^{(m+1)}(\lambda_k)$ is the pairwise effect of one of the 8-connected neighborhood.

3) Since every pixel has eight neighbors, the new $P_i^{(m+1)}(\lambda_k)$ is the average of these eight estimates as in (34).

4) Repeat steps 2)–3) until the stopping criteria is met. In our case, we specified the number of iterations in advance.

Fig. 9 shows the block diagram of our supervised textured image segmentation system. In the next section we will present the simulation results of applying the RHZ's and Peleg's algorithms to our textured image segmentation problem.

D. Simulation Results

The second test image is a texture mosaic [Fig. 10(a)] which consists of four different textures: grass, water, pigskin, and leather. All four textures are present in the image in squares of size 128 × 128. Fig. 10(b) is the gray level coded ground truth of Fig. 10(a).

We still use E5L5, E5S5, L5S5, and R5R5 as our four microtexture masks. The texture energy features are extracted by the method described in Section II. After the features are extracted, the mean vector and covariance matrix of each class are estimated by (35)–(36).

Fig. 11(a) shows the initial probability class labels for each pixel. The white pixels in Fig. 11(b) represent misclassified pixels, and the overall error rate is 3.12 percent.

We first apply the RHZ relaxation algorithm to update the probabilistic classification. Fig. 12(a) shows the classification result of 50 iterations using the mutual information coefficients as the compatibility coefficients. Fig. 12(b) is the corresponding misclassified pixels. The error rate went down from 3.12 percent to 2.4 percent as shown in Fig. 14.

Similarly, we applied the Peleg's relaxation scheme to our problem using the same initial labeling probabilities. Fig. 13(a) shows the classification result of 50 iterations using Peleg's scheme. Fig. 13(b) shows the corresponding misclassified pixels. The error rate went down from 3.12 percent to 2.06 percent as shown in Fig. 14.

From Fig. 14, we can see clearly that the Peleg's scheme converges much faster than RHZ's scheme. In ad-

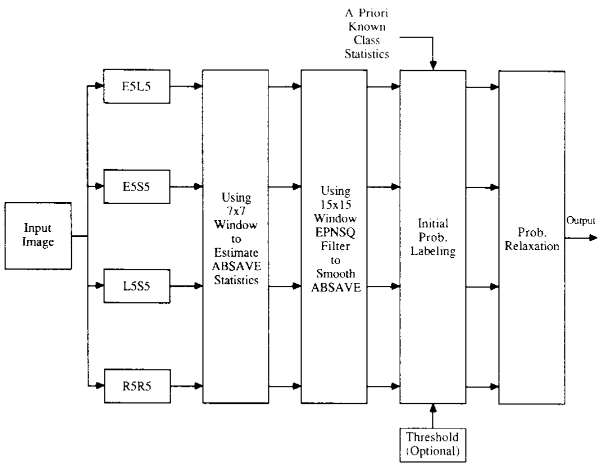

A Priori Known Class Statistics
Input Image
E5L5
E5S5
L5S5
R5R5
Using 7x7 Window to Estimate ABSAVE Statistics
Using 15x15 Window EPNSQ Filter to Smooth ABSAVE
Initial Prob. Labeling
Prob. Relaxation
Output
Threshold (Optional)

Fig. 9. Block diagram of supervised textured image segmentation system.

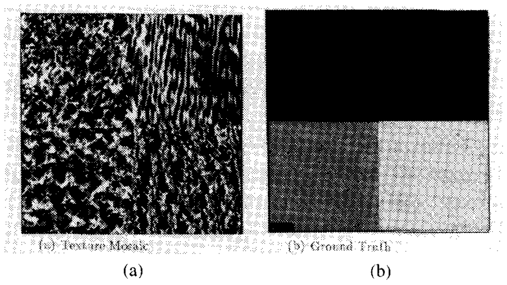


Fig. 10. Texture mosaic 2 for experimental work.

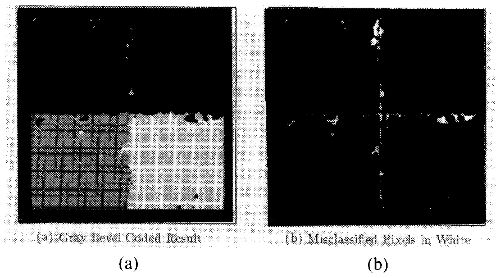

(a) Gray Level Coded Result
(b) Misclassified Pixels in White
(a)
(b)

Fig. 11. Initial probabilistic classification results.

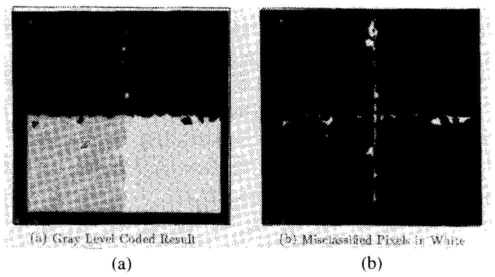

(a) Gray Level Coded Result
(b) Misclassified Pixels in White
(a)
(b)

Fig. 12. Classification result after applying RHZ's relaxation 50 iterations.

dition, both methods exhibit the desirable fact that most of the error reduction occurs in the first 50 iterations or so and then the error decreases slowly. The effectiveness

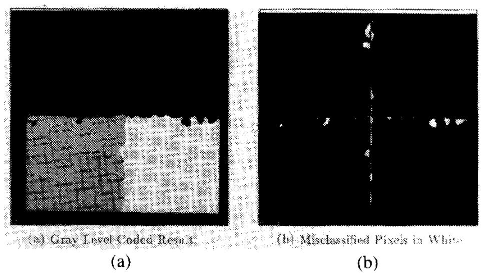

(a) Gray Level Coded Result
(b) Misclassified Pixels in White
(a)
(b)

Fig. 13. Classification result after applying Peleg's relaxation 50 iterations.

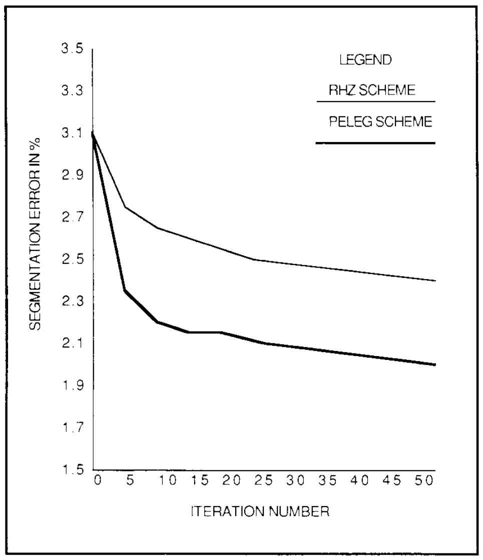
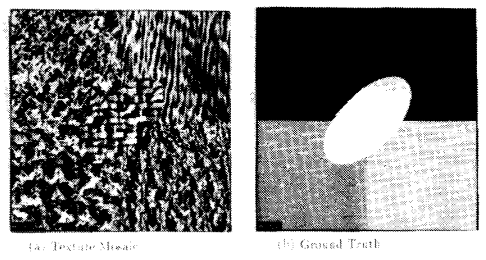


Fig. 14. The error rates of RHZ's and Peleg's scheme.

of iterative relaxation is demonstrated in Fig. 13(a), which shows that most isolated small islands in the initial classification are eliminated.

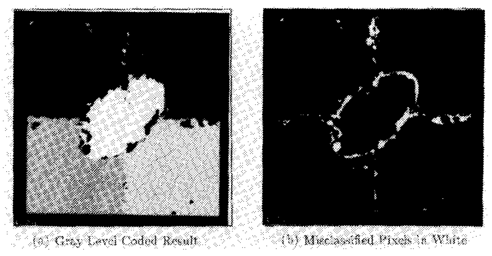
The third test image is a texture mosaic which consists of five different textures: grass, water, pigskin, leather, and raffia [Fig. 15(a)]. The fifth texture class, raffia, is presented in the middle of the image as a tilted ellipse. From the third image, we can test how well the square quadrant window shape performs on different region shapes. Fig. 15(b) is the gray level coded ground truth of Fig. 15(a).

The initial classification based on labeling pixels with the maximum initial probability is shown in Fig. 16(a). The white pixels in Fig. 16(b) indicate the misclassified ones, the error rate is 4.93 percent. The square quadrant window shape works very well on the middle region whose shape is a tilted ellipse. This result gives us an indication that as long as the region size is much larger than the size of quadrant window then the square quadrant window shape works fairly well. Fig. 17(a) shows the classification result of applying 25 iterations of Peleg's relaxation scheme. Fig. 17(b) is the corresponding misclassified pixels. The error rate went down from 4.93 percent to 3.5 percent.



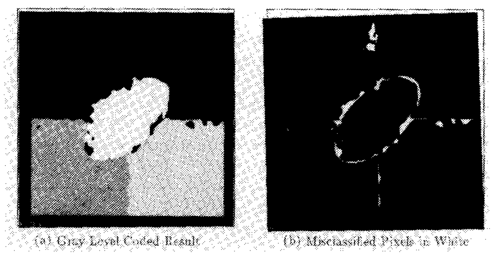
(a) (b)

Fig. 15. Texture mosaic 3 for experimental work.


(a) Gray Level Coded Result
(b) Misclassified Pixels in White

(a) (b)

Fig. 16. Initial probabilistic classification results of mosaic 3.

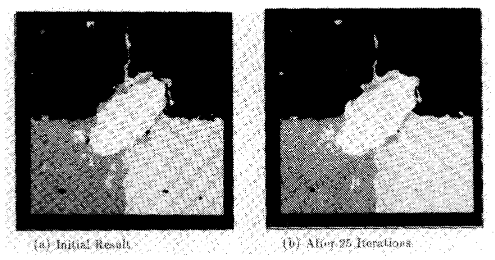

(a) Gray Level Coded Result
(b) Misclassified Pixels in White

(a) (b)

Fig. 17. Classification result after applying Peleg's relaxation 25 iterations.

As a comparison, we also process the second test image using Laws' feature extraction method as described in Section II. The window size used to estimate macrostatistical features is 15 $\times$ 15. Fig. 18(a) shows the initial classification result with an error rate of 7.2 percent. After 25 iterations, the classification result is shown in Fig. 18(b). The error rate went down from 7.2 percent to 5.4 percent. From this comparison, we can clearly see the performance is improved by using our proposed method.

In summary, the following observations can be made from the simulation results. First, probabilistic relaxation methods are an effective means to incorporate spatial constraints into segmentation algorithms. Second, probabilistic relaxation provides most of the error reduction in the first 25 iterations or so, after that the error decreases slowly. Third, Peleg's scheme is usually converges much faster than RHZ's scheme. Finally, the square quadrant window shape works fairly well on regions with arbitrary shape provided that the region size is much larger than the size of the quadrant window.



(a) (b)

Fig. 18. Classification results of mosaic 3 using Laws' method.

V. Limiting the Probability Label by Probability Thresholding

A. Introduction

One observation which we can make from the initial probability labeling process is that many pixels have very low probabilities of assignment to some of the classes. This suggests that it may be useful to limit the pixel classification only to those classes whose probabilities exceed a given threshold. To be more specific, we may define T_P to be a probability threshold, and let those initial probabilities vanish if their values are less than T_P, thus restricting the choice of possible pixel labels.

One immediate advantage of probability thresholding is that the amount of data to be processed during the following relaxation iterations decreases as a function of the threshold settings. The higher the threshold setting is, less data needs to be processed. On the other hand, once certain classes are suppressed, they will never reappear through the use of probabilistic relaxation. This becomes a tradeoff problem. Our goal is to study how different probability threshold settings affect the amount of data to be processed and its classification rates.

The algorithm we use to study the effects of limiting the decision space by probability thresholding is described as follows.

1) Choose the value of probability threshold T_P.

2) Use (35) and (36) to estimate class statistics, and then use (23) to assign initial probability labelings for each pixel.

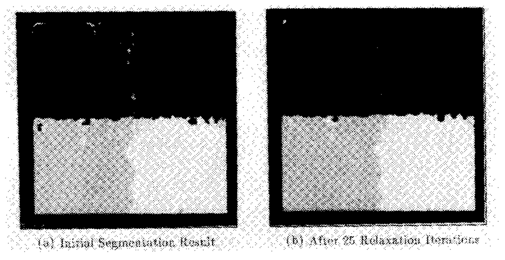
3) If any of the initial probability labeling $P_i(\lambda_k \mid \vec{x})$ is less than T_P, we set it to zero and then normalize the nonzero probabilities.

4) From these thresholded initial probability labelings, a table of compatibility coefficients may be generated.

5) A mask for each class which indicates those pixels with probability zero or one is then generated. This mask can be used to guide the relaxation process to work only on those pixels whose probabilities are not zero or one. In addition, the amount of data that needs to be processed can also be calculated from this mask.

6) With the initial probability labeling and compatibility coefficients on hand, we can start our probabilistic relaxation process.

7) After each iteration or a few iterations the classification rate may be calculated. We repeat the relaxation

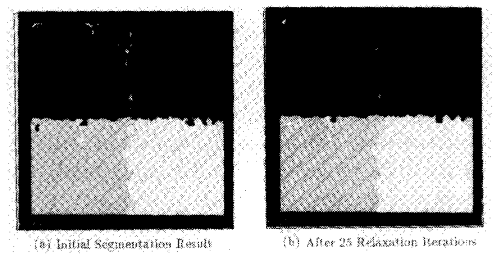

(a) Initial Segmentation Result
(b) After 25 Relaxation Iterations

(a) (b)

Fig. 19. Segmentation results of mosaic 2 with 0.005 threshold.

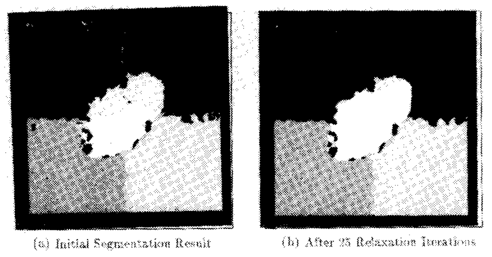
TABLE II
SUMMARY OF PROBABILITY THRESHOLD SETTINGS FOR SECOND MOSAIC

Threshold setting	Percent of total pixels to be processed	Error rate after 25 iterations (in %)
0	100	2.5
0.005	8.5	2.67
0.01	7.0	2.74


(a) Initial Segmentation Result
(b) After 25 Relaxation Iterations

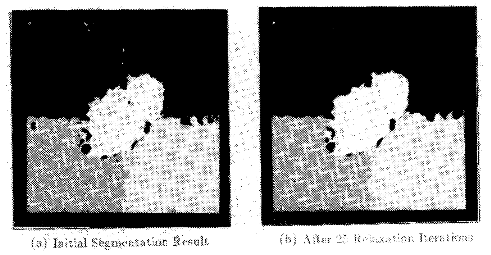
(a) (b)

Fig. 20. Segmentation results of mosaic 2 with 0.01 threshold.


(a) Initial Segmentation Result
(b) After 25 Relaxation Iterations

(a) (b)

Fig. 21. Segmentation results of mosaic 3 with 0.005 threshold.


(a) Initial Segmentation Result
(b) After 25 Relaxation Iterations

(a) (b)

Fig. 22. Segmentation results of mosaic 3 with 0.01 threshold.

TABLE III
SUMMARY OF PROBABILITY THRESHOLD SETTINGS FOR THIRD MOSAIC

Threshold setting	Percent of total pixels to be processed	Error rate after 25 iterations (in %)
0	100	3.02
0.005	5.0	3.45
0.01	4.0	3.49

process until the stopping criteria is met. In our case, we specified the number of iterations in advance.

In the next section, we present the results of applying this aforementioned algorithm with different probability thresholds to our two texture mosaics. By doing so, we can quantitatively understand the effect caused by different probability thresholds, and the results may help us in choosing the adequate threshold setting.

B. Simulation Results

Fig. 19(a) shows the initial segmentation result of the second mosaic when the probability threshold was set to 0.005 and after applying 25 iterations of relaxation to the result shown in Fig. 19(b). The total number of pixels to be processed is about 8.5 percent of the total processed if we do not set a threshold. This is a tremendous saving of computation time, however, at the expense of higher misclassification rate as shown in Table II. If we increase the threshold to 0.01, the initial segmentation and the segmentation after 25 iterations are shown in Fig. 20(a) and Fig. 20(b), respectively. The number of pixels that needs to be processed decreased from 8.5 percent to 7 percent. The results for first mosaic are summarized in Table II.

For the third mosaic we did the same experiments and the results are shown in Fig. 21 to Fig. 22. Fig. 21(a) and (b) show the results of initial segmentation and the segmentation after 25 iterations respectively for a 0.005 threshold. The number of pixels that must be processed is about 5 percent of the total pixels. Similarly, the results of initial segmentation and the segmentation after 25 iterations for 0.01 threshold are shown in Fig. 22(a) and (b), respectively. The number of pixels to be processed decreased from 5 percent to 4 percent. The results for the second mosaic are summarized in Table III.

From the above results we can draw some conclusions about the probability threshold setting. First, even a very low threshold such as 0.01 reduces the computation significantly and causes limited degradation of performance. Second, under the circumstances that the computation load is of little concern or that the clusters are not well separated in feature space then we should set the probability threshold very low (perhaps zero) in order to take full advantage of the spatial constraints. In other words, the setting of probability threshold is a data dependent problem, therefore, data analysis is needed before we can make a proper decision. For our data, the 0.01 threshold seems to be a good choice.

VI. Discussion and Conclusions

The main objectives of this work are: a) to improve segmentation results, especially along the borders of regions; and b) to take into account the spatial relationship of pixels in the process of textured image segmentation.

In order to achieve these objectives, our approach to the textured image segmentation problem is based upon the following principles:

1) The problem of estimating texture features without destroying the boundaries between regions is similar to the problem of smoothing a noisy image. Thus, techniques used to smooth noise which do not blur edges are extended to the textured image segmentation problem to improve the accuracy along region boundaries.

2) Both global information in the feature space and the spatial organization of this data in the image space must be used.

In response to the first principle, we described an algorithm which has taken the nonstationary nature of the problem into account during the feature extraction stage. We have shown that the EPNSQ filter can be applied to smooth texture feature estimates. We implemented this algorithm and tested it on a texture mosaic which consists of eight classes of textures. Based upon our simulation results, using the proposed feature estimation scheme can provide us not only better correct classification rates but also more accurate region borders for region sizes larger than 16×16. For large region sizes of our test mosaic, i.e., 128×128, the improvement on the correct classification rate is about 7 percent.

In response to the second principle, we described a supervised segmentation system which consists of two stages. The first stage assigns probabilistic labeling using the global information provided in the feature space. We realized that the weakness of classifying pixels based solely upon feature space distribution is that the formation of clusters in the feature space does not take into consideration the spatial distribution of points in the image. Therefore, we explored the use of probabilistic relaxation to reduce local ambiguities in the second stage. To the best of the author's knowledge, no previous attempts have been made in using probabilistic relaxation to solve the textured image segmentation problem. The proposed system has been implemented and tested on two texture mosaics. Based upon our simulation results, after applying 25 iterations of probabilistic relaxation about 30 percent of the misclassified pixels can be corrected which in these experiments translates into a 1.1–1.4 percent improvement on the correct classification rate.

We also used probability threshold to speed up the relaxation iterations. The proper probability threshold setting is a tradeoff between segmentation performance and speed. With some analysis of the data, the probability threshold may be set with significant improvement in speed yet with limited degradation of performance.

There are several areas for future work on this segmentation algorithm. First, even though Laws' microtexture masks are powerful in classifying textures, they were developed empirically. More study is needed to see if these masks can be derived analytically, perhaps based on statistical models, or if there is a even better set of masks. Ade [5] has proposed a set of "eigenfilters" which are derived theoretically to characterize and classify textures. The performance of eigenfilters in texture classification and segmentation remains to be seen. Unser [6] suggested that different approaches of extracting local neighborhood information by means of linear filtering operators can be unified by using a statistical vector space formulation. The statistical representation enables the definition of optimal and suboptimal linear operators for texture analysis and classification. Similar work along this line may provide an opportunity to analytically derive a set of masks with comparable or better performance. Second, for textured images with complex-shaped regions, more experimental study is needed to evaluate the behavior of the EPNSQ filter. More complicated smoothing algorithms such as the one suggested by Nagao and Matsuyama may be needed. Third, some investigation of the feature selection step which is omitted in our work is needed. For an arbitrary input textures this feature selection step may be indispensable.

A natural extension of the work presented here is to the problem in which no training data is available. This subject will be discussed in another paper [43].

Acknowledgment

We would like to thank the anonymous referees for their careful reading of a draft of this paper. Many of their valuable and constructive comments have been included in the final version.

References

[1] A. Rosenfeld and A. C. Kak, *Digital Picture Processing, Vol. 2.* New York: Academic, 1982.
[2] R. Chellappa and A. A. Sawchuk, *Digital Image Processing and Analysis, Vol. 2.* New York: IEEE Computer Society Press, 1985.
[3] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer. Vision, Graph., Image Processing*, vol. 29, pp. 100-132, 1985.
[4] J. Y. Hsiao, "Textured image segmentation using feature smoothing and probabilistic relaxation techniques," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, USC-SIPI Rep. 114, 1987.
[5] F. Ade, "Characterization of textures by "eigenfilters," *Sig. Processing*, vol. 5, pp. 451-457, 1983.
[6] M. Unser, "Local linear transforms for texture measurements," *Sig. Processing*, vol. 11, pp. 61-79, 1986.
[7] K. I. Laws, "Rapid texture identification," *Proc. SPIE*, vol. 238, pp. 376-380, 1980.
[8] K. I. Laws, "Textured image segmentation," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, USCIPI Rep. 940, 1980.
[9] R. Bajcsy, "Computer description of textured surfaces," in *Proc. Third Int. Joint Conf. A. I.*, 1973.
[10] T. Pavlidis and S. L. Tanimoto, "Texture identification by a directed split and merge procedure," in *Proc. Conf. Comput. Graph., Pattern Recogn. Data Structure*, 1975.
[11] S. G. Carlton and O. R. Mitchell, "Image segmentation using texture

and gray level," in *Proc. IEEE Conf. Pattern Recogn. Image Processing*, June 1977.

[12] T. Pavlidis and P. C. Chen, "Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm," *Comput., Graph., Image Processing*, vol. 10, pp. 172-182, 1979.

[13] ——, "Segmentation by texture using correlation," in *Proc. 5th Int. Conf. Pattern Recogn.*, 1980.

[14] H. Knutsson and G. H. Granlund, "Texture analysis using two-dimensional quadrature filters," in *Proc. ICASSP 83, IEEE Conf. Acoust., Speech, Signal Processing*, 1983.

[15] S. W. Zucker, A. Rosenfeld, and L. S. Davis, "Picture segmentation by texture discrimination," *IEEE Trans. Comput.*, vol. C-24, pp. 1228-1233, 1975.

[16] M. Pietikainen and A. Rosenfeld, "Image segmentation by texture using pyramid node linking," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 822-825, 1981.

[17] C. W. Therrien, "An estimation-theoretic approach to terrain image segmentation," *Comput. Vision, Graph., Image Processing*, vol. 22, pp. 313-326, 1983.

[18] S. Chatterjee and R. Chellappa, "Maximum likelihood texture segmentation using Gaussian markov random field models," in *Proc. IEEE Conf. Comput. Vision, Pattern Recogn.*, 1985.

[19] H. Derin and W. S. Cole, "Segmentation of textured images using gibbs random fields," *Comput. Vision, Graph., Image Processing*, vol. 35, pp. 72-98, 1986.

[20] R. W. Conners, M. M. Trivedi, and C. A. Harlow, "Segmentation of a high-resolution urban scene using texture operators," *Comput. Vision, Graph., Image Processing*, vol. 25, pp. 273-310, 1984.

[21] O. R. Mitchell, S. P. Lutton, and S. P. Su, "Texture image segmentation using local extrema," in *Proc. IEEE Conf. Pattern Recogn.*, 1979.

[22] B. J. Schachter, L. S. Davis, and A. Rosenfeld, "Some experiments in image segmentation by clustering of local feature values," *Pattern Recogn.*, vol. 11, pp. 19-28, 1979.

[23] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," *Proc. IEEE*, vol. 67, pp. 773-785, 1979.

[24] C. W. Therrien, "Linear filtering models for texture classification and segmentation," in *Proc. 5th Int. Conf. Pattern Recogn.*, 1980.

[25] F. S. Cohen and D. B. Cooper, "Real-time textured image segmentation based on non-causal Markovian random field models," in *Proc. SPIE 449, 3rd Int. Conf. Robot Vision Sensory Contr.*, 1983.

[26] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* London, England: Prentice-Hall International, 1982.

[27] R. Ohlander, K. Price, and D. R. Reddy, "Picture segmentation using a recursive region splitting method," *Comput. Vision, Graph., Image Processing*, vol. 8, pp. 313-333, 1978.

[28] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 165-177, 1985.

[29] S.-S. Jiang and A. A. Sawchuk, "Noise updating repeated Wiener filter and other adaptive noise smoothing filters using local image statistics," *Appl. Opt.*, vol. 25, pp. 2326-2337, 1986.

[30] M. Nagao and T. Matsuyama, "Edge preserving smoothing," *Comput. Graph. Image Processing*, vol. 9, pp. 394-407, 1979.

[31] F. Vilnrotter, R. Nevatia, and K. Price, "Structural description of natural textures," in *Proc. Fifth Int. Pattern Recogn. Conf.*, 1980.

[32] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[33] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover, 1966.

[34] O. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 412-424, 1981.

[35] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 420-433, 1976.

[36] S. Peleg and A. Rosenfeld, "Determining compatibility coefficients for curve enhancement relaxation processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 548-555, 1978.

[37] S. W. Zucker, E. V. Krishnamurthy, and R. L. Haar, "Relaxation processes for scene labeling: Convergence, speed, and stability," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 41-48, 1978.

[38] P. Nagin, A. Hanson, and E. Riseman, "Studies in global and local histogram-guided relaxation algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, pp. 263-277, 1982.

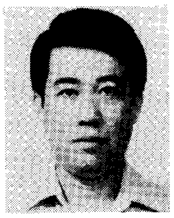
[39] S. Peleg, "A new probabilistic relaxation scheme," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 362-369, 1980.

[40] J. O. Eklundh, H. Yamamoto, and A. Rosenfeld, "A relaxation method for multi-spectral pixel classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 72-75, 1980.

[41] R. M. Haralick, J. C. Mohammed, and S. W. Zucker, "Compatibilities and the fixed points of arithmetic relaxation processes," *Comput. Graph., Image Processing*, vol. 13, pp. 242-256, 1980.

[42] R. Hummel and S. Zucker, "On the foundations of relaxation labeling processes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 267-287, 1983.

[43] J. Y. Hsiao and A. A. Sawchuk, "Unsupervised textured image segmentation using feature smoothing and probabilistic relaxation techniques," *Comput. Vision, Graph., Image Processing*, vol. 48, no. 1, 1989.



John Y. Hsiao (S'83-M'87) received the Diploma in electronics engineering from the National Taipei Institute of Technology in 1968, the M.S.E.E. degree from the University of Kansas in 1971, and the Engineer and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 1981 and 1987, respectively.

He is currently a Systems Engineer with the Advanced Programs Division of Hughes Radar Systems Group, El Segundo, CA. Since 1984, he has been with the Hughes Aircraft Company where he has been involved in developing algorithms for FLIR based Automatic Target Recognition (ATR) systems, evaluating FLIR/ATR system performance and designing multisensor fusion algorithms. Formerly, he worked at Bendix Oceanics Division, ITT Gilfillan Inc., and National Taipei Institute of Technology, Taiwan. His research interests include image processing, computer vision, pattern recognition, and signal processing.

Dr. Hsiao is a member of the IEEE Computer Society. He was a recipient of the Hughes Ph.D. fellowship.

Alexander A. Sawchuk (S'65-M'71-SM'79) was born in Washington, DC. He received the S.B. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA.

He has been employed by the National Bureau of Standards, the NASA Goddard Space Flight Center and the Communications Satellite Corporation. Since 1971 he has been with the University of Southern California, Los Angeles, where he is past Director of the Signal and Image Processing Institute and Professor in the Department of Electrical Engineering. He is also a founder and member of the Board of Directors of Optivision, Inc. His research interests include optical computing, digital image processing, machine vision, and neural systems.

Dr. Sawchuk is a Fellow of the Optical Society of America and is currently Chair of its Objectives and Policy Committee. He is a Fellow of the Society of Photo-Optical Instrumentation Engineers (SPIE) and is a member of the Optical Society of Southern California.