SRI International

GOAL-DIRECTED TEXTURED-IMAGE SEGMENTATION

Technical Note 334

September 1984

By: **Kenneth I. Laws**, Computer Scientist

Artificial Intelligence Center

Computer Science and Technology Division

SRI Project 5355

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-83-C-0027.

SRI International®

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Abstract

The SLICE textured-image segmentation system identifies image regions that differ in gray-level distribution, color, spatial texture, or other local property. It has been developed for the analysis of aerial imagery, although it can be used for any domain in which homogeneous image regions must be found prior to interpretation or enhancement. This report concentrates on textured-image segmentation using local texture-energy measures and user-delimited training regions.

The SLICE algorithm combines knowledge of target textures or signatures with knowledge of background textures by using histogram-similarity transforms. Regions of high similarity to a target texture and of low similarity to any negative examples are identified and then mapped back to the original image. This use of texture-similarity transforms during the segmentation process improves segmenter performance and focuses segmentation activity on material types of greatest interest. The system can also be used for goal-independent texture segmentation by omitting the similarity-transform computations, and its hierarchical, recursive segmentation strategy integrates very well with other image-analysis techniques.

Contents

1 Introduction 1

2 Background 2

3 Texture Transforms 2

4 Maximum Likelihood: The Trouble with Optimal Discrimination 4

5 Similarity Transforms: Encoding Goals and Knowledge 5

6 Goal-Directed Segmentation 8

7 Multiband Similarity: Putting It All Together 10

8 Examples: The Proof of the Pudding 11

9 Conclusions 19

A The PHOENIX Segmentation Algorithm 21

B Spectral Transforms for Color Segmentation 24

1. Introduction

This paper presents a new goal-directed method of textured-image segmentation. The SLICE segmentation *algorithm* is one component of a proposed knowledge-based image feature-extraction system. The algorithm is currently implemented in the SLICE *program*, a region-based recursive segmentation system running on the DARPA/DMA Image Understanding Testbed at SRI International. The SLICE program is capable of goal-independent segmentation and other image manipulations in addition to the texture segmentation discussed in this paper.

Aerial images are very difficult to segment into meaningful regions, despite the fact that humans seem to do this effortlessly. Attempts to develop segmentation algorithms using only monochrome input data have had little success. Segmentation using color and infrared data has worked somewhat better, but such data is often unavailable. This report describes techniques for using the spatial textures in monochrome imagery in much the same way that color has previously been used.

Image textures arise from many physical sources. Cellular textures are composed of repeated similar elements, such as leaves on a tree or bricks in a wall; other texture types include flow patterns, fiber masses, and stress cracks. A complete *analysis* of any texture would require modeling of the underlying physical structures and processes. In most applications, texture *recognition* is more important than knowledge of the generating mechanism. The algorithm presented in this report can be used for texture recognition and material identification when we have knowledge of scene texture types, and for locating and characterizing textured regions even when we have no such knowledge.

The SLICE algorithm consists of three parts: goal-directed texture transformation, multiple histogram-based threshold segmentations, and spatial analysis of the proposed segmentations in order to choose the best one. These steps may be repeated on the newly found regions to further segment them. A high-level control system could be used to focus the segmenter's "attention" on important image regions and can determine when to stop partitioning a given region (using size, shape, homogeneity, semantic, or priority considerations). Regions found by other image-analysis techniques can also be combined with those found by the SLICE algorithm.

This report describes the SLICE algorithm and the rationale for each part of the technique. Section 2 introduces some definitions used throughout the report. Section 3 briefly describes the basic texture transforms used to measure local spatial variation around a pixel. Section 4 discusses maximum likelihood classification methods, and points out why they are not optimal for texture segmentation. Section 5 presents similarity transforms that can be used to locate desired texture signatures in an image. Sections 6 and 7 describe the integration of texture similarity transforms with histogram-based segmentation to produce goal-directed segmentation using multiple texture bands. Section 8 then

presents examples of the technique, and Section 9 summarizes the characteristics of this approach. Details of the modified PHOENIX goal-independent color-image segmentation technique used in the current SLICE program are presented in Appendices A and B.

2. Background

An image is a two-dimensional array of pixels, where pixels are numbers (usually integers in the range 0 to 255) or vectors of numbers representing information about an imaged scene. An image of vector-valued pixels may be thought of as a set of two-dimensional, scalar-valued layers called data bands. (Indeed, the pixel data is usually stored in this layered fashion.)

Pixel values typically represent intensity of light (infrared, visible, or ultraviolet) or other electromagnetic energy reaching a sensor from a point in the imaged scene, but may correspond to other measurable scene properties. Data bands may also record such computed information as stereo disparity, intensity gradients, filter responses, estimated scene albedo, or inferred surface slope. In this paper we shall be particularly concerned with texture bands computed from local texture properties around each pixel.

The integer values that can be assumed by a scalar pixel are called gray levels. Even nonphysical data values such as texture statistics are representable as gray levels or intensities, since the data bands may be displayed on an image monitor as black-and-white luminance images. A special type of data band, called a *map*, stores at each pixel an integer value representing the material type or other nominal category assigned to that pixel. A *segmentation map* or *region map* has a unique integer assigned to all pixels in an image region and different integers assigned to different regions. Segmentation maps are often displayed using pseudocolor (i.e., arbitrary assigned region colors), since display as a luminance image is not meaningful.

The value of a pixel is easily read out of the array; all other information about the imaged scene is implicit. It is the task of low-level image processing to make useful spectral or spatial information explicit so that high-level feature-extraction operators and reasoning processes can utilize it. This paper will describe a goal-directed method for extracting homogeneous image regions satisfying prespecified criteria as to size, location, gray-level distribution, and texture.

3. Texture Transforms

Textures can be recognized if one or more distinctive properties can be measured.[1] Many ways of computing texture descriptors have been proposed. Some of the most powerful descriptors, both individually and in combination, are the *texture-energy* measures [Laws 80] and their variants [Pietikäinen 82, Harwood 83]. The transforms require only simple convolution and moving-window updating techniques; moreover they can be made invariant to changes in image illumination, contrast, and orientation without histogram equalization or other preprocessing operations.

Texture energy is the amount of variation within a filtered window around a pixel. A particular texture-energy measure thus depends on the spatial filter, the window size, and the method of measuring average variation within the window. These measures do

[1]There are also structural or "syntactic" pattern-recognition methods that do not require texture metrics.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

$L3 * L3$ $\qquad$ $L3 * E3$ $\qquad$ $L3 * S3$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix}$$

$E3 * L3$ $\qquad$ $E3 * E3$ $\qquad$ $E3 * S3$

$$\begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$S3 * L3$ $\qquad$ $S3 * E3$ $\qquad$ $S3 * S3$

Figure 1: Orthogonal 3×3 Texture Masks

not describe texture-generating mechanisms or parameters directly, but do tend to be constant across any perceptually homogeneous texture region and distinct for distinct textures. (Within macrotextures having large elements they tend to be multimodal with histogram peaks corresponding to the edges and interiors of the texture elements.)

The first step is to filter the original scalar image with a small convolution mask. The mask is typically a binomially weighted array (defined below) that enhances image spots, edges, or high-frequency components. Binomially weighted masks are both separable and decomposable into smaller convolution masks, making them easy to implement efficiently on a variety of architectures. The set of filter masks used determines the spatial frequencies or texture structures that the transforms will measure.

The 3×3 binomially weighted masks are shown in Figure 1. They were constructed by convolving the vectors [1 2 1], [−1 0 1], and [−1 2 − 1] with their own transposes.[2] Larger masks may be constructed by convolving the 3×3 masks with themselves. Binomially weighted filter masks of sizes 3×3, 5×5, and 7×7 were found to be nearly equivalent in tests on a very limited class of textures [Laws 80]. For aerial image analysis, the 3×3 masks seem likely to be the most useful, although strong patterns such as orchards and crop rows may be better discriminated using larger masks.

An exponential series of mask sizes may be needed for multiresolution texture analysis. Application of large binomial weighted masks (with coefficients in the billions) can be

[2]These vectors are themselves constructed from the vectors [1 1] and [−1 1]. The mask names are derived from the terms *level*, *edge*, and *spot* for the 3-vectors of sequency 0, 1, and 2. Similar names are used for the 5-vectors and 5×5 masks, with the addition of W (*wave*) and R (*ripple*) for the vectors of sequency 3 and 4.

very difficult even if done by repeated filtering with smaller masks. A better method is to construct a pyramid of image reductions and then apply a single mask size to all levels of the pyramid. The unfiltered image itself may be used as the highest-resolution "filtered" band, and its local-energy statistics may be useful either as texture measures or for normalizing the other texture measures when contrast invariance is desired.

The second texture-transformation step is to apply a *local-energy* operator to the filtered image to produce a texture-energy data band. Texture energy at a point is just the variance of the filtered-image values computed over a window around the point. Standard deviation, or the square root of the variance, has been found just as effective. For zero-mean filtered bands, the standard deviation is usually approximated by an average of the filtered-image magnitudes (i.e., absolute values) over a window. Such averages can be computed by moving-window techniques that are very fast, even on general-purpose digital computers.

An energy-gathering window of about about five or ten times the area of the filter mask is recommended; larger sizes give better classification accuracy when applied to large texture patches but lack the resolution needed for analysis of typical 512×512 aerial image displays. The time required to compute the local energy is independent of the window size, since each pixel is examined only once as it enters the window and once as it leaves. (A fading-memory approximation to moving-window averaging can also be used to permit single-pass computation of texture energy values without storing the intermediate filtered image.)

Texture descriptors computed with the suggested masks are unaffected by most scene illumination and sensor bias effects because all but the first mask produce zero-mean outputs. Level-invariant texture energy bands may be normalized by the variance of the unfiltered image if contrast-invariant texture measures are desired. Pairs of texture energy values representing directional image structure can also be averaged if orientation-invariant texture measures are desired. Decisions about when to normalize or average are best left to a control system capable of reasoning about particular analysis tasks and image contexts.

4. Maximum Likelihood: The Trouble with Optimal Discrimination

We now have a multiband image composed of luminance or spectral bands and derived texture bands. We want a quick way to partition the vector-valued pixels into homogeneous groups, preferably using a priori knowledge of target signatures when it is available. The texture transforms make it likely that each texture signature in the image will have a fairly predictable Gaussian distribution in at least one data band. The temptation to jump to multivariate Gaussian discriminant analysis is almost overwhelming.

There is a good reason for trying other methods, however, even when we have sufficient multivariate training data to compute the needed means and variances (or covariance matrices). Maximum-likelihood Gaussian discriminant analysis[3] is optimal for separating two or more multivariate Gaussian distributions, but we do not have Gaussian distributions as such—we have mixtures thereof. Even within a single data band we may have at best one Gaussian and one mixture density to be discriminated.

[3]I.e., multivariate minimum-distance classification using an inverse-covariance, or *Mahalanobis*, weighting.

This is not to say that the discriminant analysis won't work, only that the conditions for optimality are not satisfied. The procedure for computing discriminant functions will reduce positive and negative training instances to means and standard deviations, reject any data bands in which the means and standard deviations are similar, and do the best it can with a linear or perhaps quadratic function of the means and standard deviations in the remaining bands. The result is that much of our knowledge about target signatures in different data bands will be discarded.

As an example, consider a texture that is known to have a Gaussian distribution in a particular data band. Assume that the scene might also contain instances of another texture with a strongly bimodal salt-and-pepper distribution. These two distributions should be easily distinguished, but they may not be discriminable by mean and standard deviation alone. We could use multivariate statistics and compute covariances with data values in other bands, but the resulting classifier might be unstable and difficult to train. We could also seek a transform to another data band in which the two textures are separable, but that will not work if there are other textures that might also be present in the scene. Discriminant analysis is thus a poor way to deal with this situation.

There is also the matter of a priori probability. Any form of maximum-likelihood classifier performs best if the decision thresholds are properly adjusted for the a priori probability of that texture's appearing in the scene. We may be able to guess reasonable probabilities based on previous analyses of similar imagery, but biasing the analysis with such values instead of just examining the evidence in the image is a suspect procedure. Assigning equal likelihood to all possible scene entities is even more suspect.

We could live with these problems, but there are better ways of finding and distinguishing textures. These will be described in the remainder of this paper.

5. Similarity Transforms: Encoding Goals and Knowledge

The key to efficient goal-directed segmentation is to estimate quickly whether any given pixel is part of the texture or target signature we are seeking. We have already computed texture bands in order to collapse implicit information about a pixel's neighborhood into explicit information stored in the pixel's data vector. We now need a scalar measure of similarity (or, conversely, of dissimilarity) between a pixel data vector and a target signature.

The simplest dissimilarity measure is the Euclidean distance between the image pixel vector and a prototype vector representing a known texture type or previously extracted region signature. We could invert this if we wanted a similarity measure. We could also weight the component single-band distances differently if we had knowledge that some data bands were more critical for recognition than others.

The easiest way to determine, or to "learn," which data bands are important is to keep track of the multivariate statistics within a target population and compare them with the statistics for other possible scene entities. This leads to Mahalanobis distance as a measure of dissimilarity between a pixel and a prototype. As discussed above, this would be optimal for Gaussian distributions, since they are fully characterized by their means and covariances.

It is not necessary to represent a texture prototype by statistical vectors and matrices, nor is it necessary to specify complex parsing rules. An intermediate strategy

is to represent a texture or target signature by its full histogram in each data band. (A knowledge-based system would also have rules for manipulating these histograms in accordance with overall image illumination and contrast; we shall assume here that any required normalization has been done or will be compensated for during the image analysis.)

Storing histogram vectors as prototypes is very easy for a region-based system because the region histograms are always readily available. To train the system one has only to trace or extract a suitable region, assign it a label, and store it in the knowledge base. The only complication is that different data bands may be used during different image analyses, so that prototypes saved during one session may not include the bands needed during another session. There would also be some difficulty if the same data band were scaled or quantized differently for each session, but we can usually compensate for such discrepancies.

Computing similarity between a pixel and a prototype is a bit more difficult than computing Mahalanobis distance. We can split the problem into that of computing similarity within a given band and that of combining different band similarities into a single overall similarity measure, although such a two-step procedure is not necessarily optimal.

Consider then the problem of estimating how likely it is that an observed gray level in a data band came from one prototypical population and not from another. We may formalize and generalize this problem as follows. Given that we have observed a gray level g as an independent random sample, what is the probability that the source population was one of a set ω of positive exemplar distributions, ω_i, and not from one of a set ϕ of negative exemplar distributions, ϕ_j? We shall assume that we know each of the prototype distributions by a single histogram representing an unbiased sample from some large population. We have, then, sets of histograms estimating $\Pr(g|\omega_i)$ and $\Pr(g|\phi_j)$ and we wish to compute $\Pr(\omega|g)$.

If we assume disjoint and exhaustive source populations, we can compute the probability that the observed sample gray level g came from a positive exemplar source population as the normalized sum of probabilities for the members of the set:

$$\Pr(\omega|g) = \frac{\sum_i \Pr(\omega_i|g)}{\sum_i \Pr(\omega_i|g) + \sum_j \Pr(\phi_j|g)} \quad .$$

The denominator should sum to one, given our assumptions, but this formula will work even if the probabilities are expressed relative to some larger set of disjoint source populations.

Bayes' rule applied to each component term in the summations gives us

$$\Pr(\omega|g) = \frac{\sum_i \Pr(g|\omega_i)\Pr(\omega_i)}{\sum_i \Pr(g|\omega_i)\Pr(\omega_i) + \sum_j \Pr(g|\phi_j)\Pr(\phi_j)} \quad ,$$

where a term $\Pr(g)$ has been canceled from the numerator and denominator. This is the theoretical form of the similarity function that we need. If we assume equal a priori probability for each source population, the formula simplifies to

$$\Pr(\omega|g) = \frac{\sum_i \Pr(g|\omega_i)}{\sum_i \Pr(g|\omega_i) + \sum_j \Pr(g|\phi_j)} \quad .$$

Although the current SLICE program makes this simplification, additional knowledge of the scene domain might provide a better set of weightings.

Now, how do we compute $\Pr(g|\omega_i)$? We could simply take the bin count for bin g in the ω_i histogram and divide it by the total number of counts in the histogram. This would have two undesirable effects: the estimated probability for adjacent bins could vary wildly because of sampling fluctuations or "picket-fence" quantization effects, and the similarity formula could not be evaluated for bins that happened to be empty in all prototype histograms.

If the histograms were samples from Gaussian distributions, we could use the sample mean and variance to estimate the true population bin probabilities for every gray level. Since we generally have mixture densities, this approach would require that every prototype texture histogram be decomposed into component Gaussians. While this is difficult, it could be done (at least approximately) either automatically or interactively at the time a texture prototype is entered into the system's knowledge base. We note that this is an optimal solution, but will now proceed to develop a much simpler heuristic approximation.

If a distribution is known to be Gaussian, we achieve the greatest predictive power by using techniques appropriate to that parametric form. If we have no knowledge of the parametric form, we can still treat the histogram as a sample taken from a multinomial distribution having unknown bin probabilities.

An observed bin probability, $\Pr(g|\omega_i)$, is an unbiased estimate of the true bin probability in the sampled population. It is not, however, the best estimate of that generating probability, given that a sample has been taken. An example may clarify this somewhat difficult concept. Suppose that we have formed our prototype histogram by sampling a single pixel. We shall then have a single populated bin and 255 empty bins (assuming 8-bit quantization). Are we then willing to say that our best estimate for the population distribution is a spike at the observed gray level and zero probability of any other value? No, we would wish to be more conservative, even if we were not assuming an underlying Gaussian model.

Our intuition serves us well here. Bayes showed in 1763 that the a posteriori probability of a given multinomial bin-generating probability, given an observed histogram, has a beta distribution, which is a continuous distribution resembling a skewed binomial [Jaynes 83]. The mean, or expectation, of this distribution is [Abramowitz 64, p. 930]

$$\frac{\text{observed bin count} + 1}{\text{number of bins} + \text{number of samples}}$$

Using this as our $\Pr(g|\omega_i)$ has the effect of smoothing the population histogram estimate slightly by adding a fractional count to each bin. This permits us to compute our similarity measure even for bins that are to be empty in all prototype histograms.

The similarity function is now optimal for multinomial distributions, but not for Gaussian mixture densities. It fails to allow either for the strong correlation between nearby bin counts that is due to the component densities or for the exponential decrease in bin frequency as a gray level is chosen farther from any histogram peak. The first effect is particularly noticeable when the training data contain regularly spaced empty bins resulting from a sticky quantizer bit or from contrast stretching that introduced a picket-fence envelope. While we can imagine separating two textures by their differing picket-fence characteristics (i.e., by trivial differences in gray levels), this is not the type of behavior we want to build into our image segmenter.

The solution, short of actually finding the component Gaussian densities, is simply

to smooth the histograms. The SLICE program uses a binomial kernel (this is the best discrete approximation to a Gaussian) with a standard deviation of 1, 3, or 5 pixels. Histogram counts are scaled by 1000 so that fractional bin counts can be represented; this scaling must be compensated for when computing the similarity transform.

The above smoothing extends each tail of a histogram peak for a dozen pixels or so, then drops to zero. The multinomial bin correction that is subsequently applied will lift this slightly above zero, but by an amount that does not vary with distance from the histogram peak. This causes undesirable behavior of the similarity transform for gray levels near the ends of typical histograms. Consider the case of a very sharp Gaussian peak for our positive exemplar and a broad peak or mixture density for our negative one. Further assume that the positive-exemplar histogram contains only a few hundred pixels and that our negative exemplar is based on a very large sample, typically the entire image we wish to segment. We would expect that image pixels far from the positive exemplar peak would have very low similarity to that texture type because the associated Gaussian distribution would have a very sharp exponential decay. Instead we compute a high similarity because the multinomial correction for a histogram with few counts is a much larger number than that for a histogram with many.

This leads to one more adjustment, a factor that provides exponential (i.e., Gaussian) decay in the multinomial correction as we select bins farther from the nearest or broadest peak in a histogram. There is no need to be precise here, so we can use an approximation based on the distance to the lowest or highest count in the smoothed histogram. Only the multinomial correction is applied for gray levels that are between these two limits. For pixels outside the observed sample range, the 1 in the multinomial correction is replaced by $\exp(-\frac{2d}{w})$, where d is the distance to the nearest bin in the observed range and w is the number of bins in that range.

6. Goal-Directed Segmentation

We now have a rapid method of computing the similarity of an observed gray level in an image band to a set of positive exemplar textures (or target signatures) with respect to a set of negative exemplar textures. We can apply this function efficiently by precomputing the similarity measure for every possible texture-band gray level and then looking up each image pixel's value in the resulting table of values. We shall usually have a particular texture or target signature as a positive exemplar and shall use the full image (or region) histogram as a negative one. (This implicit inclusion of the texture we are seeking in the negative-exemplar histogram will not cause problems unless it is a major component of that histogram. If it is, we might first suppress that component of the negative-exemplar histogram by subtracting a multiple of the positive-exemplar histogram. The SLICE program does not yet include such a correction procedure.)

Our problem, then, is to select a similarity threshold that will separate all (or at least most) of the instances of our target texture from instances of all other textures. If multiple similarity bands are available, we should either select the best band for our threshold segmentation or combine the information in all the bands. This section describes a segmentation method capable of selecting the best similarity band for extracting examples of the target texture and of recognizing those cases in which no satisfactory threshold can be found. The next section will discuss other methods of combining information from

multiple similarity bands.

For any texture band, the computed similarity value for each pixel should ideally be near 1.0 for the texture we are seeking and near 0.0 for the textures specified as negative training examples. The actual separation for any real data band will be less, and some texture bands may fail to discriminate the training textures at all, but a decision threshold at 0.5 should separate our positive and negative training textures if they are indeed discriminable. Decision thresholds above or below 0.5 could also be chosen; this is equivalent to adjusting the a priori source-class probabilities, $\Pr(\omega)$ and $\Pr(\phi)$, that are implicit in the similarity transform. (We can no longer second-guess the relative proportions of the population probabilities, $\Pr(\omega_i)$ and $\Pr(\phi_j)$, within the source class probabilities. Such fine control is not needed, however, particularly since we rarely require multiple positive or negative exemplars.)

"Unexpected" or unmodeled textures in the image should have similarity values in between the extremes for the training textures. The [eight-bit] histogram of a similarity band typically has a very large peak at the low-similarity end (representing image gray levels that were common in the negative-exemplar textures) and a spread of higher-similarity spikes that look rather uniformly distributed. Smoothing the histogram (with a Gaussian kernel of standard deviation 5 or less) typically reveals that this high-similarity energy consists of a few Gaussian clusters representing image regions that are fairly similar to the positive exemplar.

Segmenting this smoothed histogram is usually quite easy. The SLICE program currently uses a version of the PHOENIX histogram-based segmentation algorithm (documented in Appendix A) to find suitable thresholds. We may want to select just the peak of values most similar to the training texture, although thresholding anywhere above the large peak of least-similar values will generally produce a good spatial segmentation of the image. We may also select multiple thresholds that will isolate other peaks in the histogram and thus extract image regions with other textures as well. It is this capability that makes SLICE a segmentation algorithm rather than just a classification algorithm.

The above procedure works if it is possible to find even one peak in one histogram that is reasonably well separated from other peaks. In practice, there can be as much as a 25% overlap between two Gaussian peaks and the segmenter will still find reasonable subregions in the image. There will be times, of course, when the histogram-segmentation algorithm fails to find any segmentable peaks in the similarity-band histogram, particularly when we are trying to segment a whole black-and-white image or a low-resolution texture data band. The segmenter will find that a region is uniform and unsegmentable, but higher-level knowledge may suggest that this is false. The current SLICE program is not able to proceed automatically in such cases, but any of the following techniques could be invoked:

- Try again with relaxed parameters for the peak-finding heuristics. The SLICE program currently uses the PHOENIX histogram-partitioning heuristics with the "moderate" parameter settings developed during the SRI evaluation of that package for the DARPA/DMA Image Understanding Testbed. These smoothing parameters and heuristic criteria could be successively weakened until peaks are found in the histogram.

- Compute additional data-band transformations such as pairwise ratios or combinations of existing data bands. Any oblique cut through the multidimensional his-

togram space is likely to resolve at least one histogram peak. Computation of such a data band and histogram does not take long, particularly if only a small region is involved.

- Compute a multidimensional histogram from multiple data bands and apply cluster analysis techniques to find discriminable subpopulations. Combining two bands in this way produces a two-dimensional histogram that can be analyzed by means of image partitioning techniques [Nagin 77, 78]. The SLICE system itself might be used to find populated areas of the bivariate histogram.

- Threshold the image region at arbitrary levels, e.g., at histogram quartiles or deciles, and use spatial analysis (including noise suppression) to recover subregions that can later be remerged or edited. This option is available in the SLICE program and works surprisingly well.

- Partition the region at arbitrary spatial boundaries, segment the pieces, and then remerge or edit subregions along the boundaries [Robertson 73, Horowitz 74, Price 76].

- Switch to a different histogram-segmentation method, such as minimal-spanning-tree analysis or relaxation-based peak sharpening [Bhanu 82].

- Switch to an entirely different segmentation approach, e.g., region growing from homogeneous seed areas within the region.

Any of these methods, and no doubt others, will partition the image into regions containing fewer homogeneous pixel populations. Once partitioning is started, subregions that are themselves composite can usually be segmented with ease.

Once distinct histogram peaks have been found, the segmentation algorithm finds corresponding regions in the image by simple threshold segmentation and connected-component extraction. It then computes a quality score for the spatial segmentation based on the percentage of small "noise regions" produced. This quality score can be used to select the best of several competing similarity-band segmentations. Better quality scores could be computed from region shapes and other high-level or goal-directed criteria.

The current SLICE program includes an optional screening of extracted regions based on spatial adjacency. Frequently the user wishes to "grow" an identified image region (e.g., one that he has traced with a pointing device) instead of finding all other similar pixel patches in the image. After connected-component extraction, the SLICE program can suppress any region that is not touching or nearly touching the initial prototype. Connected components are again extracted and the analysis proceeds. The final step in the growing process is to merge the regions found with the original training region.

7. Multiband Similarity: Putting It All Together

The previous section described a method for finding image regions corresponding to peaks in a similarity-band histogram. The implemented segmentation algorithm is able to select the best of several competing similarity bands by comparing the identified histogram peaks and quality of spatial segmentation produced by each set of similarity-band thresholds. This approach is typically useful in cueing applications when searching a scene

for textures that might differ considerably from stored prototypes. Using this technique, a target region distinguishable in even one data band can be segmented from its background and passed up to a higher-level reasoning system for confirmation. The method also works well with multiple data bands containing essentially the same information, since slight differences in the information content might lead to better segmentation in one band than in the others.

Another approach, also available in the current SLICE program, is to combine the similarity bands computed from different luminance or texture bands into a single overall similarity band. This is appropriate when we are very sure that our prototypes are representative, as when we are trying to find a homogeneous texture region around a traced seed region. Under these conditions we can assume that all instances of the target texture will look very similar to the training texture in all transformed bands—if a subregion differs significantly in even a single band it cannot be from the target population.

We might use factor analysis or discriminant analysis to devise an optimum weighting function for combining the similarity bands. Such a function would no doubt be task-dependent and image-dependent, making it very difficult to assemble sufficient training data. A simpler solution is to construct the composite similarity band from the pixel-by-pixel minima of the component similarities. This combining function is often used in fuzzy-set theory. It correctly reflects the assumption that target textures should behave just like the prototype texture under any transformation, but has the negative effect that we cannot recover from a prototype that is unrepresentative in even a single data band.

In practice, the use of similarity minima works quite well for region growing. When combined with spatial analysis and noise cleaning it results in either a reasonable segmentation or a failure to segment at all. Other combining functions might work better in particular cases, however. Higher-level guidance based on preliminary segmentation and analysis of each similarity band could be used to choose combining functions intermediate between the first all-or-nothing approach and the second pixel-by-pixel minima approach [Salton 83, Rauch 84].

One of the consequences of the SLICE similarity band computation is that the inclusion of additional similarity bands in a composite should never degrade performance (other than taking longer to compute). If we have truly representative prototypes, any histogram-based transformation will either help discriminate the positive training classes from the negative training class or it will fail to do so. If it discriminates them partially, any reasonable combining function will either make use of the information or, at worst, ignore it; if it does not discriminate them, the corresponding similarity band will be essentially constant and will do no harm. (The SLICE program currently tests for such useless similarity transforms and does not bother to compute the similarity band.) The SLICE similarity-transform approach thus has the advantage that it may fail to provide information but will seldom produce misinformation.

8. Examples: The Proof of the Pudding

We shall now examine the performance of the SLICE algorithm on a particular aerial image analysis task. The processing sequence will demonstrate both the advantages and the disadvantages of this approach. Simple ways of improving the demonstrated performance will also be discussed.

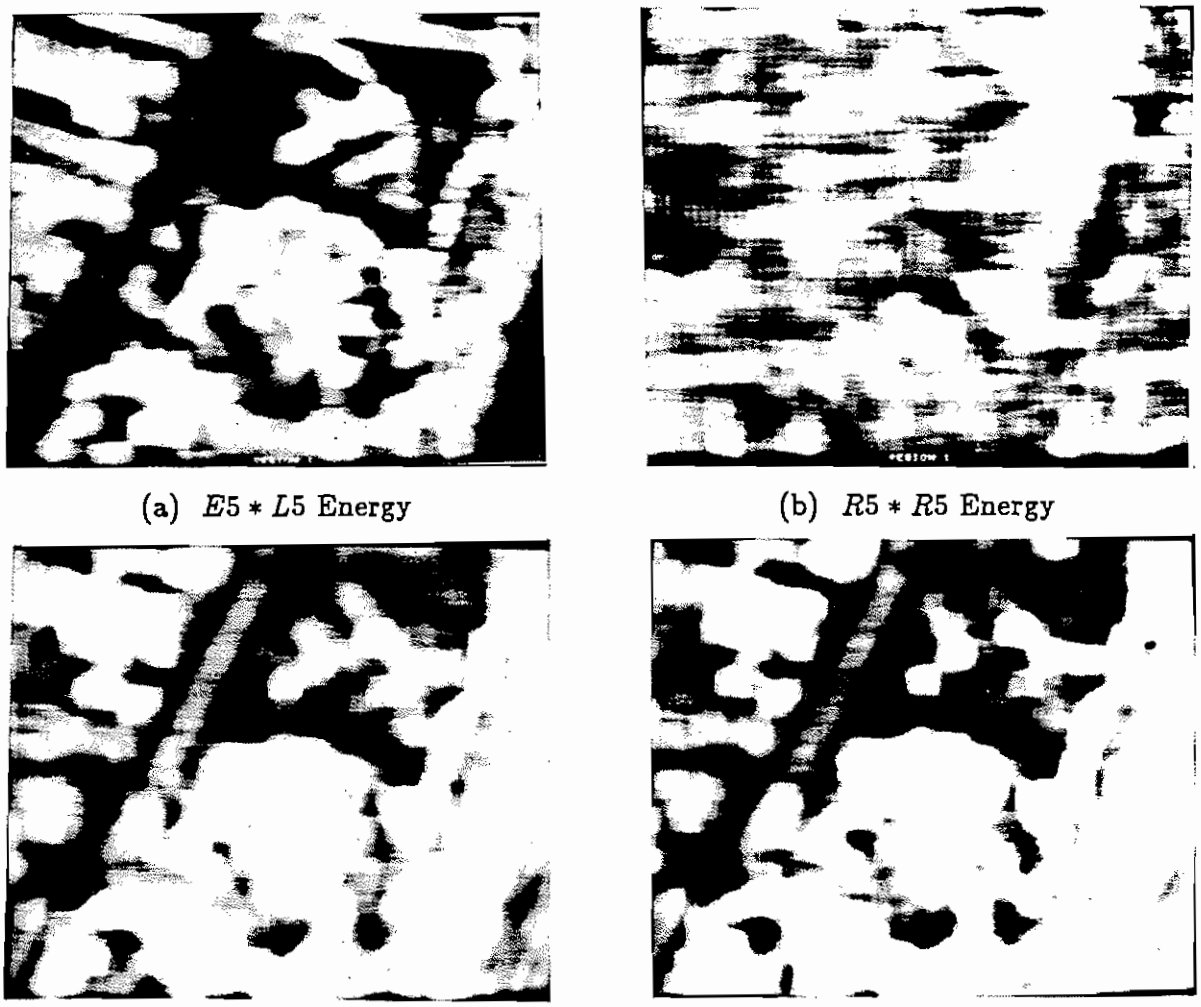
Figure 2: Aerial Image with Positive and Negative Training Regions

Figure 2 is a black-and-white image of a residential area near Page Mill Road in Palo Alto, California. The image has not been normalized or otherwise preprocessed. It shows trees, roads, fields, buildings, swimming pools, and a few cars. Shadows are cast both by the buildings and by the trees, although the resolution is such that tree shadows are very difficult to discern. A full parse of the scene would detect and identify all of these entities. Our concern here will be the extraction of all the tree regions, perhaps as a preliminary to extracting other objects in the scene. We shall first trace the process of "growing" tree regions from user-selected examples and shall then examine the output of a more general "finding" or cueing algorithm.

Extraction of the trees in this image by interactive threshold segmentation is not difficult. The trees (or perhaps bushes) are distinguishable by their gray-level signatures in the original image. An image-understanding system would not know this a priori, however, but would have to extract and identify at least some of the trees and then estimate whether it could extract the rest. The system could search for good tree regions by experimentation with different thresholds [Selfridge 82], but the methods presented in this paper are more efficient.

Trees presumably have distinctive texture signatures in addition to their gray-level signatures. Figure 3 shows four texture bands selected from the texture energy set. These were computed with 5×5 filter masks and 15×15 "absolute average" summations. (At this image resolution, the 3×3 filter set would probably have worked better.) Each texture band highlights different characteristics of the original image. The four texture bands used here are the same ones selected in the development of the texture energy measures [Laws 80], although there is little reason to believe that this is the best subset or even an adequate subset for image analysis. The selected filters respond primarily to horizontal edges, high-frequency variation, medium-frequency diagonal structure, and narrow or high-frequency vertical structure. The data bands have been normalized for image contrast, but pairs have not been combined to form orientation-independent texture bands.

Goal-independent analysis of the original image, using the PHOENIX segmentation



(a) $E5 * L5$ Energy

(b) $R5 * R5$ Energy

(c) $E5 * S5$ Energy

(d) $L5 * S5$ Energy

Figure 3: Texture Energy Bands

algorithm (and skillful setting of its numerous parameters), leads to isolation of most trees in the scene. These regions are not the first to be reported by the segmenter, however, nor are they the last. The user or high-level control program must somehow select the tree regions from among the hundreds of reported regions. This is made more difficult by the fact that most of the scene is very poorly segmented by the PHOENIX algorithm, with region boundaries crossing homogeneous fields and with parts of house roofs cut off and grouped with surrounding fields. Adding computed texture bands to the original black-and-white band degrades performance: areas around building edges are identified as homogeneous regions and several scattered patterns of trees interspersed with grass are also extracted as regions. (The latter effect is useful, but not as useful here as finding the individual trees.)

Goal-driven segmentation with the SLICE algorithm begins with the selection of training areas. A sophisticated system might have adequate tree templates stored in its knowledge base. Here we depend on the user to select representative training regions. Large samples work better than small ones, but we will demonstrate the technique with the two small training areas in Figure 2. A negative training region containing a strong shadow and a mixture of other scene textures is also shown; the remainder of the image outside the three traced areas will be used as a second negative training region.

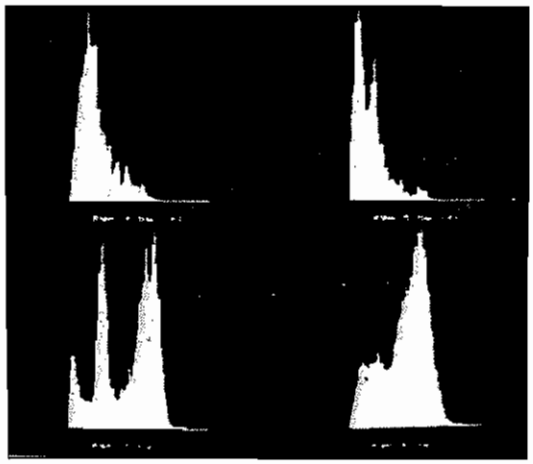
The two positive training regions were carefully selected. The upper-right region is a nearly minimal sample such that the SLICE program's "grow" command will extract the entire clump of trees extending from the upper-right corner diagonally downward toward the bottom edge of the second training region. The second region is also a nearly minimal sample such that the "grow" command will extract all tree clumps touching the region. Rather than witness these feats, we will now examine performance when both regions are sought simultaneously and the upper negative training example is also specified.

Figure 4(a) shows the histograms of the training regions together with the histogram of the remainder of the image. The two similar histograms at the top of Figure 4(a) correspond to the two user-traced training regions containing trees. The bottom-left histogram corresponds to the negative training region; it has three peaks corresponding to a shadow, the house roof plus driveway, and the lawn and car. The bottom-right histogram for the rest of the image (also used as a negative training example) contains some dark tree and shadow pixels and many light pixels from other scene components.

Figure 4(b) shows the resultant gray-level similarity transform for this black-and-white image band. Similarity is highest for those pixels corresponding to trees, despite the negative effect of tree regions in the whole-image histogram. The similarity transform shows a very slight dip for shadow pixels and a much stronger dip in the house roof interval. The least-similar gray levels are those occurring in the image but not in either positive training region. The similarity function increases again for very bright pixels: these are absent in all image regions and hence need not concern us.

Applying this transformation to the original image produces the similarity data band in Figure 4(c). Trees, shadows, and a very small number of other scene objects have been highlighted. We can easily extract trees from this transformed band, and such a segmentation will be presented in Figure 8. For now, let us continue trying to "grow" our training regions using multiband similarity.

The SLICE program is capable of using any number of texture bands as additional inputs. For demonstration purposes, we shall limit it to just the band of Figure 3(b). This

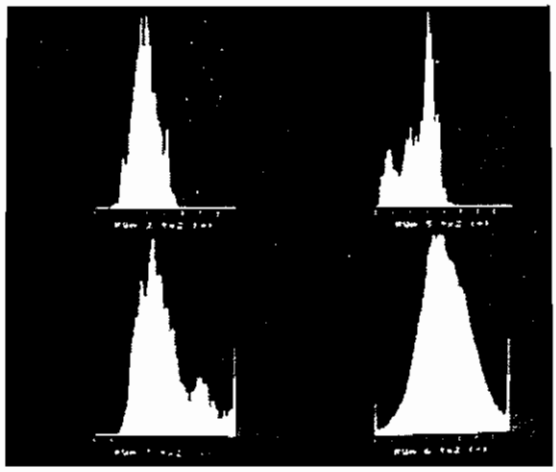


(a) Training Histograms

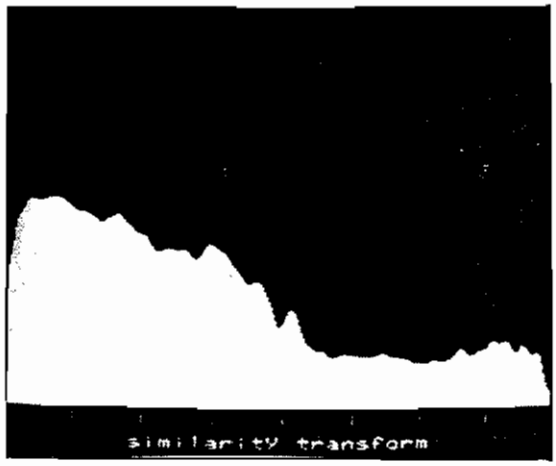
(b) Similarity Transformation

(c) Similarity Band

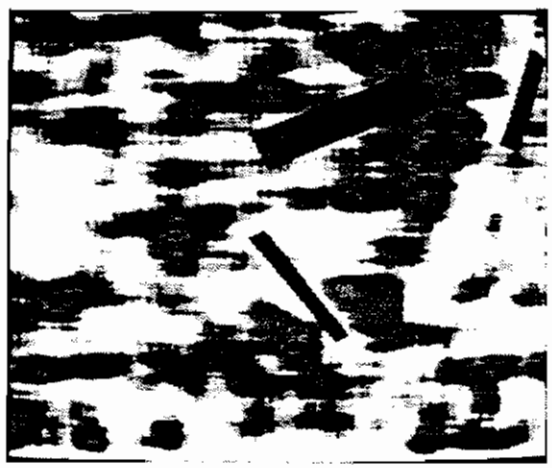
Figure 4: Original Image Analysis



(a) Training Histograms

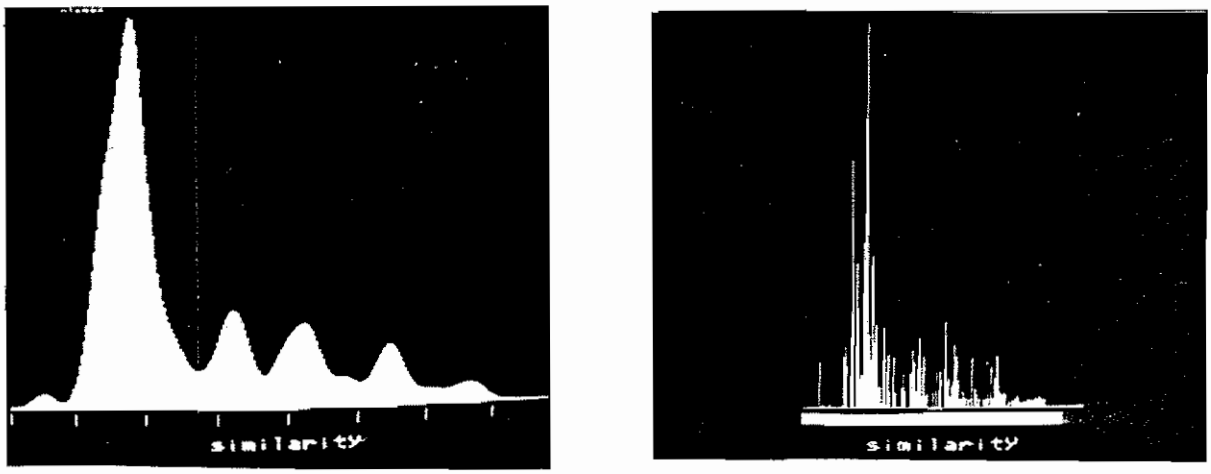


(b) Similarity Transformation



(c) Similarity Band

Figure 5: $R5 * R5$ Texture Band Analysis



(a) Smoothed (b) Unsmoothed

Figure 6: Merged Similarity-Band Histogram

band tends to show trees as dark regions, although the effect is not strong. Any of the other texture bands might perform as well or better.

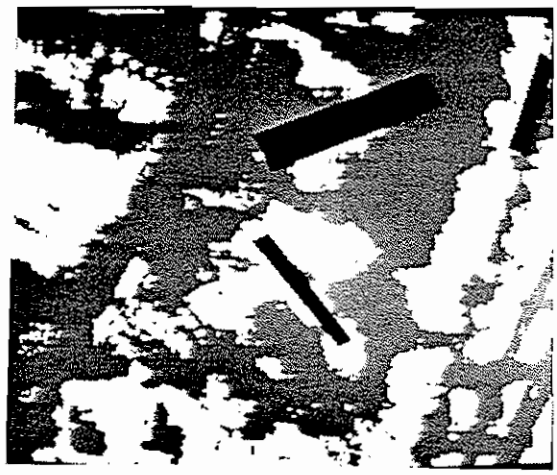
Figure 5(a) shows the two positive training histograms (top) and two negative training histograms (bottom) for this band. Note how strongly the histogram on the bottom left matches that of the positive training areas. Use of this negative training example actually degrades segmenter performance in this case, albeit only slightly. (In other situations the "caution" introduced by this overlap might prevent the segmenter from making bad decisions.) We could reduce the degradation by giving the negative example less weight than the histogram of the area to be segmented, but the knowledge-based mechanisms needed to make such decisions have not been included in the SLICE program.

Figure 5(b) shows the similarity transform computed from the training histogram. It shows a preference for dark pixels, but is not very specific. This leads to the texture-similarity band of Figure 5(c), which we can see will not lead to a good segmentation of the image. The segmentation program has no such perception, however, and must somehow determine that it should reject most of the "information" in this computed band.

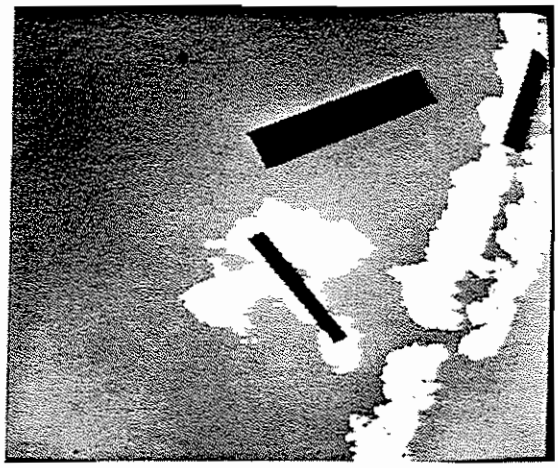
The method of combining similarity functions that we will use here is to take the pixel-by-pixel minimum of all similarity bands. This combined transform function can be computed from the individual similarity transformations rather than from the similarity data bands, thus saving considerable computation. The result of applying this combined similarity transformation to the original image may be seen in Figure 7(a). It is similar to the similarity band for the black-and-white data band, although more intermediate gray levels are present.

Figure 6 shows both the smoothed and the unsmoothed histograms of the merged similarity band. There are several clusters in this one-dimensional space, and any to the right of the main histogram peak could represent the trees we are seeking. An intelligent system would investigate several thresholds or would use several thresholds simultaneously. The current SLICE algorithm simply chooses the threshold that best survives its screening heuristics—in this case perhaps a rather poor choice. Figure 7(b) shows the spatial result of applying this threshold. The trees are indeed found, but so are driveways, road patches,

(a) Similarity Band



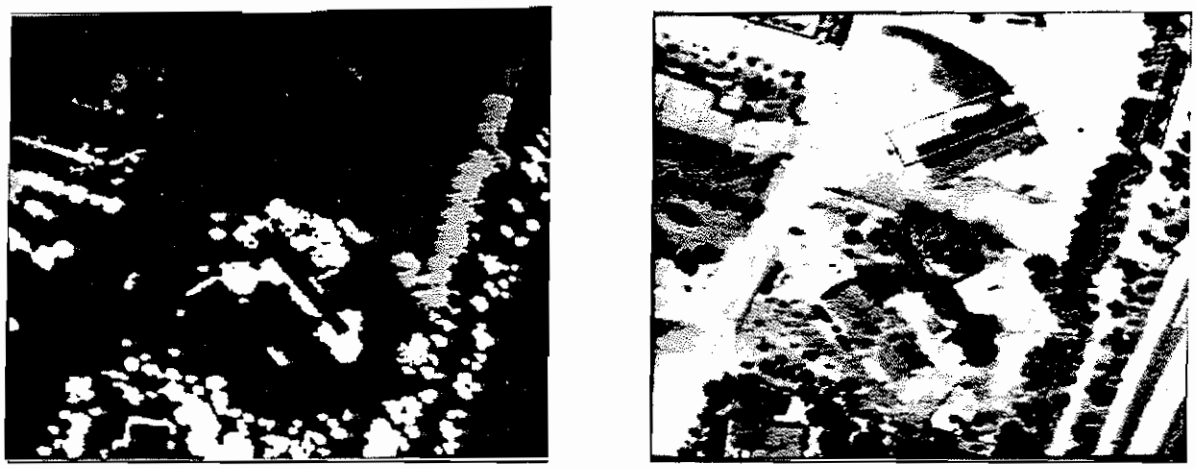
(b) Thresholded Band



(c) Extracted Regions

(d) Region Outlines

Figure 7: Merged-Similarity-Band Analysis



(a) Gray-Level-Coded Region Map (b) Region Outlines

Figure 8: Regions Found by Similarity-Band Selection

and other dark image regions. There has also been a blurring effect because of the 15×15 window used to compute the texture energy band.

Our current task is to "grow" the original sample regions to their full image extent, rather than find more distant matching regions. A simple spatial analysis can thus be employed to eliminate all regions not touching the original training regions. (The current criterion is that the rectangle enclosing a candidate region must come within one pixel of touching a rectangle enclosing one of the positive training regions. This allows for small breaks in our extraction of a scene object and permits a higher-level process to determine whether nearby regions should be merged or discarded.) Candidate regions smaller than some threshold, here set at five pixels, are also discarded as probable noise regions.

Figures 7(c) and 7(d) show the result of this spatial screening. The segmenter has done a good job of expanding the initial sample regions, although the lower-left region has absorbed part of an adjacent house roof. The upper-right tree clump has been found, but includes pixels from the surrounding field that would not have been included if the inherently blurred texture band had not been used (or if the final similarity histogram had been thresholded at a higher gray level). Two additional large regions, both containing trees, are found because of the gray-level and spatial interactions of the two sample regions. They would not have been retained if the two training regions had been grown independently.

This concludes the presentation of the goal-directed region-growing technique used in the SLICE program. We have seen how the algorithm is able to overcome difficulties such as small positive training regions; negative training regions that are unrepresentative (i.e., badly weighted) and include the very pixels we are trying to find; blurred, poorly chosen, or uninformative texture data bands; ad hoc similarity combining functions; and poorly chosen thresholds. As knowledge-based techniques are developed and refined, some of these difficulties will be eliminated and performance enhanced commensurately.

A final example of the power of the SLICE goal-directed approach may be observed in Figure 8, which shows the effect of the SLICE program's "find" command when the same training regions and texture band are employed. This algorithm differs from region

growing only in the similarity combining function and spatial screening. Instead of combining the computed similarity bands, each is analyzed separately and the one with the least "noise area" is selected. The texture data band is thus rejected and only the original black-and-white image is employed. Shadows and a few other undesired dark areas are found, but essentially every tree over five pixels in area is identified. This leaves very little work for a higher-level verification process to perform.

9. Conclusions

The SLICE segmentation system is one of several existing systems for segmenting digital images recursively. Its major contributions are computation of a nearly optimal texture-similarity function and integration of this approach with a robust segmentation system to permit both goal-directed and goal-independent textured-image partitioning. Some of the advantages and disadvantages of the SLICE algorithm are listed below.

- The SLICE goal-directed segmentation algorithm uses multispectral or "multitextural" input to extract precisely those scene objects of most interest. If it fails, repeated attempts with relaxed constraints may locate candidate regions. If it succeeds, it generally produces high-quality regions that require little postediting. Provision for negative training examples permits easily confused material types to be separated early in the analysis process.

- SLICE, like other region-based methods, always yields closed region boundaries. This is not true of edge-based feature extraction methods, with the possible exception of boundary following and zero-crossing detection. Closed boundaries are not needed for all image-analysis tasks, but they do simplify many approaches. In particular, the resulting regions provide meaningful entities for a human or high-level control system to reason about and manipulate.

- SLICE is a hierarchical or recursive segmenter, which means that even a partial segmentation may be useful. This can save a great deal of computation if efforts are concentrated on image regions in which further segmentation is critical. If a full goal-independent segmentation is desired, however, other methods of segmenting may be more economical.

- SLICE is relatively insensitive to noise because noise tends to average out in the region histograms used to select thresholds. This contrasts with edge-based methods, as the local analysis they require can be highly perturbed by noise. The SLICE program also eliminates small noise patches by merging them with surrounding regions during connected-component extraction, but a knowledge-based decision function for identifying noise patches could improve this process.

- SLICE currently has no notion of boundary straightness or smoothness. This may be either good or bad, depending on the scene characteristics and the analysis task. It easily extracts large homogeneous regions that may be adjacent to detailed, irregular regions (e.g., a lake adjacent to a dock area or the sky over a complex skyline); such tasks can be difficult for edge-based segmenters. Boundary aesthetics and other semantic criteria can be incorporated as part of either an editing process or knowledge-based control structure.

- Region-based segmenters may fail to detect even long and highly visible boundaries between two large, similar regions if the region textures cause their histograms to overlap. The use of texture bands may reduce this problem because the boundary region itself forms a distinctive texture. Hypothesis-driven edge-based methods (e.g., construction of specific-orientation edge detectors optimized for the textures on either side) may be required to confirm such boundaries.

- SLICE tends to miss small regions within large ones because they contribute so little to the composite histogram. It is thus poorly suited to goal-independent detection of vehicles and small buildings in aerial scenes, although the use of multiple texture bands alleviates this deficiency. Goal-dependent selection of search areas and texture-similarity transforms will help to locate small objects even against backgrounds of similar gray level.

- SLICE also tends to misplace the boundary between a large region and a small one, thus obscuring roads, rivers, and other thin regions. Boundaries found by edge-based methods are less affected by distant scene properties, but work poorly if not adapted to the statistics of the regions being discriminated. An edge-based postediting of the region boundaries found by SLICE may combine the best of both approaches.

- SLICE requires multispectral input or multiple texture transforms for effective operation. Edge-based and valley-seeking or spanning-tree techniques are better adapted to operation in a single data band, and thus require less computer memory and possibly less processing time.

Selection of a segmentation algorithm should depend on the task to be performed. The SLICE segmentation system is a convenient testbed for integrating diverse feature-extraction techniques and experimenting with knowledge-based control structures.

Acknowledgment

This research was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903–83–C–0027.

Appendices

A. The PHOENIX Segmentation Algorithm

The SLICE segmentation algorithm incorporates the PHOENIX color segmentation algorithm developed at Carnegie-Mellon University [Shafer 82]. The PHOENIX algorithm is a sophisticated method of hierarchical region extraction based on region statistics and user-specified parameters. It does not use explicit knowledge about the types of data bands it is given nor about the scene objects being sought. This section describes the operation of the PHOENIX algorithm. The reader wishing further description and details of operation is referred to the SRI Image Understanding Testbed evaluation of the PHOENIX system [Laws 82].

Each object or object part in a scene is assumed to form a nearly uniform patch in the image, with a noisy Gaussian peak in any single-band histogram. Decomposing a function into Gaussian peaks is known as the mixture density problem [Wolfe 70] and is important in information theory, statistics, chemistry, and other fields. Very little of this theory has been applied to image processing [Chow 70, Rosenfeld 76, Postaire 81]. The PHOENIX/SLICE algorithm segments mixture densities by identifying the most obvious thresholds in any of the data bands, then using spatial-analysis "look-ahead" before confirming a candidate threshold. The algorithm does make slight errors in threshold placement, however, leading to the breakup of some small regions and a shifting of the boundaries of others.

Ohlander and Price used a hierarchy of heuristic rules for selecting the most prominent peak within a set of histograms [Ohlander 78, Price 79, Nevatia 82]. PHOENIX uses similar heuristics, but concentrates on the valleys (i.e., local minima) in the histogram set. Usually a single valley, resulting in one threshold and two intervals, is selected for each feature. Spatial analysis is then employed to select the best threshold/data band combination. Using only one threshold per pass reduces the chance of segmentation errors, although it does increase the number of passes required.

Histograms can be treated as one-dimensional images and can be segmented by almost any image segmentation method. The PHOENIX histogram-analysis component uses an *interval-merging* strategy. Each single-band histogram is first smoothed with a binomial or Gaussian smoothing kernel having a standard deviation gsmooth— typically 3, and ranging from 5 for coarse segmentation down to 2 for more detailed segmentation. (The original PHOENIX algorithm used a simpler unweighted moving average.) The histogram is then broken into intervals in such a manner that each begins just to the right of a valley (i.e., at the next higher intensity), contains a peak, and ends with the next valley. A valley is considered to be the right *shoulder* of its left interval and the left shoulder of its right interval. The leftmost and rightmost intervals always have exterior shoulders of zero height.

A series of heuristics is then applied to screen out noise peaks. Each test is applied to all the intervals in the histogram. When an interval is eliminated, it is merged with the neighbor sharing the higher of its two shoulders. The screening test is then applied again to the merged interval. (Previous tests are not reapplied.)

Peak-to-shoulder ratio is tested first. An interval is retained only if the ratio of peak height to the higher of its two shoulders, expressed as a percentage, is at least as great as the user-supplied **maxmin** parameter—typically 160%, and ranging from 300% for "strict" screening down to 130% for "mild" screening.

Peak area is then compared with an absolute threshold, **absarea**, and with a relative threshold, **relarea**, representing a percentage of the total histogram (or region) area. Only peaks larger than these thresholds are retained. **Absarea** is typically 30 pixels, ranging from 100 pixels down to 5 pixels; **relarea** is typically 2%, ranging from 10% down to 1%.

The intervals surviving to this point should be reasonable candidates, and it is fairly safe to use global histogram descriptors in the test conditions. The second-highest peak is now found, and those peaks whose height is less than a percentage, **height**, of it are merged. The lowest interior valley is then found, and any interval whose right shoulder is more than **absmin** times that valley height is merged with its right neighbor. (The parameter appears to be misnamed, since the criterion is relative rather than absolute.) Typical values of these parameters are 20% and 10 pixel counts, ranging from 50% to 10% and 2 counts to 30 counts.

A final screening is performed to reduce the interval set to **intsmax** intervals. This is done by repeatedly merging regions with low peak-to-shoulder ratios until only **intsmax**−1 valleys remain. **Intsmax** is typically set to 2 to force the highest-quality segmentation during each pass, although higher values could save considerable computation time.

A score is also computed for each interval set as a whole (in relation to the interval sets for other data bands). This score is the maximum over all intervals of the function

$$1000\,\frac{\text{peak height} - \text{higher shoulder}}{\text{peak height}}$$

This formula assigns the maximum score to an interval set containing a peak with shoulders of zero height. Interval sets with scores less than **absscore** or less than **relscore** percent of the best score for all data bands are rejected. **Absscore** is typically 700, ranging from 925 down to 600; **relscore** is typically 80%, ranging from 95% down to 65%.

If more than **isetsmax** data bands are still candidates for segmentation, the excess ones with the lowest scores are now dropped. This parameter is typically 3 and ranges from 2 to 5. Remaining data bands and interval sets are passed to the spatial-analysis subsystem.

Histogram segmentation is a heuristic technique that sometimes misses good thresholds and sometimes chooses bad ones. Some protection is provided by examining segmentations of several different data bands and choosing the best. Regions smaller than the **noise** threshold are merged back into their parent regions and bands producing region segmentations with more than **retain** percent of their area so merged are rejected. These parameters are typically 10 pixels ranging from 50 pixels down to 5 pixels, and and 20% ranging from 4% to 40%. The remaining segmentation producing the lowest noise percentage is then selected and instantiated in the data base. All resulting subregions are scheduled for further attempted segmentation provided that their areas are at least **splitmin** pixels—typically 40 pixels and ranging from 200 pixels down to 20 pixels.

No single threshold is going to result in perfect segmentation when the histogram peaks overlap. We might instead use two thresholds—one low enough to catch all of the higher peak and another high enough to catch all of the lower peak—then ascertain from the

image which threshold is correct for extracting each subregion. In practice, most of the small *noise patches* that result from a slightly offset threshold are easy to identify and absorb into the surrounding subregions. The noise-cleaning process leaves only the exact placements of the subregion boundaries in doubt, and these can be better determined in a postediting of adjacent region pairs than through clever partioning of a multiregional histogram.

B. Spectral Transforms for Color Segmentation

Color bands are needed when two regions to be distinguished have similar texture (including intensity), but different hue or saturation. Transformations of these bands can sometimes be used to separate pixel clusters that project to overlapping or confounded histogram peaks in the original spectral data bands. Similar band combinations may be useful for segmenting texture bands or other nonspectral data bands.

Color transformations are not currently implemented as part of the SLICE program, but transformed data bands computed off-line can be used to improve its operation. The segmentation algorithm currently makes no distinction between color bands and other types of data bands, although the associated display routines do make such a distinction.

Color bands for image processing research are typically generated by scanning a color photograph through filters (e.g., Wratten filters 25, 47B, and 58) to get red, green, and blue (RGB) data bands. Real-time systems often use an electronic color camera to generate equivalent YIQ[4] bands that correspond roughly to perceptual brightness, cyan vs. orange, and magenta vs. green. The following discussion assumes that the primary input is in RGB coordinates, but converting to or from other color coordinates is fairly easy.

Each color system constitutes a three-dimensional chromatic space that can express most of the colors perceived by humans. (The detailed spectrum that astronomers and other physical scientists depend upon has been lost, just as it is in the human visual system.) A few purples and highly saturated colors are not precisely representable and the colors recorded with different films or cameras may differ, but the tricomponent representation is adequate for most purposes.

Typical quantization is eight bits per color axis, or 16.8 million cells for an entire three-dimensional color histogram. Cluster analysis in such a space is not attractive, although methods of multidimensional pattern recognition are available. The SLICE package instead uses an adaptation of a one-dimensional histogram partitioning method implemented in the CMU PHOENIX program [Tomita 73, Tsuji 73, Ohlander 75, Shafer 82, Laws 82].

Any one-dimensional histogram is equivalent to a projection of the three-dimensional data onto a line (or curve) through the chromatic space. If the scene contains many regions, their histogram peaks are likely to overlap and obscure any useful details in the composite histogram. The overlap is different for projections at different angles, and it is often possible to isolate peaks from some of the regions by using many projections.

Ohlander used RGB, HSD[5], and YIQ projections, but many other color coordinate transformations are possible. The HSD coordinates were introduced by Tenenbaum et al.

---

[4] YIQ is the National Television Systems Committee (NTSC) color coordinate system. The perceptual brightness, or Y, chromaticity band takes its name from the XYZ chromatic primary system of the Commission Internationale de l'Eclairage. I and Q are the NTSC *in-phase* and *quadrature* signal components.

[5] The HSD, or hue–saturation–intensity, color coordinate system is also known as the HSI or IHS system. The symbol D is used here for intensity to avoid confusion with the YIQ system. It comes from *density*, a measure of the amount of silver deposited at a given point in a photographic negative.

[Tenenbaum 74a, 74b] to mimic human color perception. They are

$$
\begin{aligned}
H &= \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)(R-G)+(R-B)(G-B)}} \\
S &= m \cdot (1 - 3\frac{\min(R,G,B)}{R+G+B}) \\
D &= \frac{(R+G+B)}{3} \quad ,
\end{aligned}
$$

where m is the maximum desired saturation value. Hue is normalized by subtracting it from 2π if $B > G$; some care must be taken in rounding the values near 2π if the number is quantized. Note that these formulas contain singularities that are due to division by zero, and thus exhibit unstable segmentation behavior near the D axis.

The YIQ coordinates used in color television transmission are

$$
\begin{aligned}
Y &= 0.509R + 1.000G + 0.194B \\
I &= 1.000R - 0.460G - 0.540B + M \\
Q &= 0.403R - 1.000G + 0.597B + M \quad ,
\end{aligned}
$$

where M is the highest possible intensity value in the original RGB features, typically 255. These formulas have been linearly scaled to maintain quantization accuracy (via the unit coefficient). M is added simply for convenience in digital representation. (The Q feature can be negated before adding M to better match the green gun on a color monitor.)

Kender analyzed the color transformations used by Tenenbaum and Ohlander and showed that inherent singularities and quantization effects were capable of introducing false histogram peaks and valleys [Kender 76, 77]. This effect is particularly noticeable in the hue feature, but also affects saturation and other normalized chromaticity coordinates. He recommended that saturation be ignored in regions of low luminance, with hue ignored in low saturation as well. The YIQ transform was found to entail fewer problems, although its usefulness in segmentation was not evaluated. Kender also proposed an improved computational algorithm for hue.

Ohta et al. have further investigated color transforms for recursive segmentation [Ohta 80a, 80b]. They computed color histograms by using the Karhunen-Loeve color transform—an expensive method because the transform is different for each region. Ohta found that the principal transform axes typically clustered around

$$
\begin{aligned}
I_1 &= R+G+B \\
I_2 &= R-B \\
I_3 &= 2R-(G+B)
\end{aligned}
$$

and recommended that these features be employed. (The second and third features may be negative, so that either an offset becomes necessary or the segmentation code must be able to handle negative pixel values.)

Ohta's transform is similar to the YIQ system, as well as to the opponent-color (i.e., red-green and blue-yellow) representation recommended by several authors [Sloan 75, Nagin 78]. The transform is linear and hence avoids the instabilities that Kender found in

saturation, hue, and normalized chromaticity coordinates. Nagin expressed some theoretical reservations about his own opponent features, but concluded that they "consistently provided more discrimination than the original RGB data."

HSD and YIQ color transformations were used in SRI's evaluation of the PHOENIX color segmentation program [Laws 82]. Hue was mapped to the range 0 to 179, with red at 0 (and 180), green at 60, and blue at 120. Achromatic pixels (i.e., black, gray, and white) were mapped to 255; this seldom makes a difference since pixels with exactly equal RGB components are exceedingly rare. A less exact test for achromaticity might work better (or at least differently) for images with slight imbalances in their color strengths.

The I and Q color bands computed by Kender's formulas should theoretically be divided by two (and then shifted to a nonnegative range) if they are to be stored in 8-bit image planes. (The SLICE program can handle image data with other pixel sizes, but eight bits is convenient and seems to offer a reasonable dynamic range.) Most of this range is wasted, however, unless I is stretched by a factor of two and Q by a factor of four prior to quantization, with clipping of extreme values. This greatly increases the usefulness of these bands for segmenting natural imagery, although it could fail for scenes that contain large regions of saturated colors.

Hue was generally not only the most useful color band in the SRI evaluation, but also the easiest to comprehend. The D and Y bands are essentially redundant; they do not always segment identically, but the extra information is not worth the effort of computing both. Segmentation on the RGB bands was almost as good but more difficult to explain; the RGB bands were each nearly equivalent in segmenting power, and successive region extractions seemed to jump randomly from one to another. (Differences in color are usually correlated with disparities in brightness, so an object that appears red might actually be segmented on a different color band by the PHOENIX/SLICE algorithm.) The S band was somewhat less useful, although decisions based on it were easy to explain. I and Q were the least useful data bands, although they might have been essential if some of the other seven data bands had not been available.

Overall, the HSI color system seems easiest to use, although the other color systems work well if explanations of each segmentation step are not needed. The RGB bands are so similar to one another that the addition of a hue band can improve segmentation greatly. Pixels containing blue mixed with red (i.e., purples and violets) are rare even in hazy mountain scenes, so there is seldom a problem with peaks in the hue histogram being split between the bottom and top portions of the scale. Saturation is more likely to be the cause of such instabilities; dark or shadowed image regions sometimes transform to very high saturation values, indicating that segmentation on luminance should be done first or that the instability of saturation should be considered during noise cleaning and other analyses.

Transformations of texture and other nonspectral data bands have not been evaluated, but simple sums and differences of the bands are the most separated in a multidimensional histogram space and are thus most likely to improve segmenter performance. Texture bands can also be combined with spectral bands in this way.

References

[Abramowitz 64] M. Abramowitz and I.A. Segun (eds.), *Handbook of Mathematical Functions,* National Bureau of Standards (1964). Reprinted by Dover Publications, Inc., New York, New York (1965).

[Bhanu 82] B. Bhanu and O.D. Faugeras, "Segmentation of Images Having Unimodal Distributions," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI–4, No. 4, pp. 408–419 (July 1972).

[Chow 70] C.K. Chow and T. Kaneko, *Boundary Detection of Radiographic Images by a Threshold Method,* IBM Research Report RC–3203 (1970). Also in *Proc. IFIP 71,* TA–7, p. 130 (1971), and in S. Watanabe (ed.), *Frontiers of Pattern Recognition,* Academic Press, New York, New York, pp. 61–82 (1972).

[Harwood 83] D. Harwood, M. Subbarao, and L.S. Davis, *Texture Classification by Local Rank Correlation,* TR–1314, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland (August 1983).

[Horowitz 74] S.L. Horowitz and T. Pavlidis, "Picture Segmentation by a Directed Split-and-Merge Procedure," *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition,* Copenhagen, Denmark, pp. 424–433 (13–15 August 1974).

[Jaynes 83] E.T. Jaynes, "Where Do We Stand on Maximum Entropy," in R.D. Rosenkrantz (ed.), *Papers on Probability, Statistics, and Statistical Physics,* D. Reidel Publishing Co. (1983).

[Kender 76] J.R. Kender, *Saturation, Hue, and Normalized Color: Calculation, Digitization Effects, and Use,* Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (November 1976).

[Kender 77] J.R. Kender, "Instabilities in Color Transformations," *IEEE Conf. on Pattern Recognition and Image Processing,* Troy, New York, pp. 266–274 (6–8 June 1977).

[Laws 80] K.I. Laws, *Textured Image Segmentation,* Ph.D. Thesis, Report USCIPI 940, USC Image Processing Inst., University of Southern California, Los Angeles, California (January 1980).

[Laws 82] K.I. Laws, *The PHOENIX Image Segmentation System: Description and Evaluation,* Technical Note 289, Artificial Intelligence Center, SRI International, Menlo Park, California (December 1982).

[Nagin 77] P.A. Nagin, A.R. Hanson, and E.M. Riseman, *Region Extraction and Description through Planning,* Computer and Information Science Report 77–8, University of Massachussets at Amherst (May 1977).

[Nagin 78] P.A. Nagin, *Segmentation Using Spatial Context and Feature Space Cluster Labels,* Computer and Information Science Report 78–8, University of Massachussets at Amherst (May 1978).

[Nevatia 82] R. Nevatia, *Machine Perception,* Prentice-Hall, Englewood Cliffs, New Jersey (1982).

[Ohlander 75] R. Ohlander, *Analysis of Natural Scenes,* PhD. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburg, Pennsylvania (April 1975).

[Ohlander 78] R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing,* Vol. 8, No. 3, pp. 313–333 (December 1978).

[Ohta 80a] Y. Ohta, T. Kanade, and T. Sakai, "Color Information for Region Segmentation," *Computer Graphics and Image Processing,* Vol. 13, pp. 222–241 (1980).

[Ohta 80b] Y. Ohta, *A Region-Oriented Image-Analysis System by Computer,* Ph.D. Thesis, Dept. of Information Sciences, Kyoto University, Japan (March 1980).

[Pietikäinen 82] M. Pietikäinen, *Image Texture Analysis and Segmentation,* Ph.D. Thesis, Department of Electrical Engineering, University of Oulu, Oulu, Finland, Acta Universitatis Ouluensis, Series C, Technica No. 21, Electronica No. 1 (February 1982).

[Postaire 81] J.G. Postaire and C.P.A. Vasseur, "An Approximate Solution to Normal Mixture Identification with Application to Unsupervised Pattern Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI–3, No. 2, pp. 163–179 (March 1981).

[Price 76] K.E. Price, *Change Detection and Analysis in Multi-Spectral Images,* Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (December 1976).

[Price 79] K.E. Price, "Segmentation," *Proc. IEEE Conf. on Pattern Recognition and Image Processing,* Chicago, Illinois, pp. 512–514 (6–8 August 1979).

[Rauch 84] H.E. Rauch, "Probability Concepts for an Expert System used for Data Fusion," *The AI Magazine,* Vol. 5, No. 3, pp. 55–60 (Fall 1984).

[Robertson 73] T.V. Robertson, P.H. Swain, and K.S. Fu, *Multispectral Image Partitioning,* TR–EE 73–26, LARS Information Note 071373, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana (August 1973).

[Rosenfeld 76] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC–6, No. 6, pp. 420–433 (June 1976).

[Salton 83] G. Salton, E.A. Fox, and H. Wu, "Extended Boolean Information Retrieval," *Communications of the ACM,* Vol. 26, No. 12, pp. 1022–1036 (December 1983).

[Selfridge 82] P.G. Selfridge, *Reasoning about Success and Failure in Aerial Image Understanding,* Ph.D. Thesis, TR103, Computer Science Dept., University of Rochester, New York (May 1982).

[Shafer 82] S. Shafer and T. Kanade, "Recursive Region Segmentation by Analysis of Histograms," *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing,* Paris, France, pp. 1166–1171 (May 1982).

[Sloan 75] K.R. Sloan and R. Bajcsy, "A Computational Structure for Color Perception," *Proc. ACM '75,* Minneapolis, Minnesota (1975).

[Tenenbaum 74a] J.M. Tenenbaum, T.D. Garvey, S. Weyl, and H.C. Wolf, *An Interactive Facility for Scene Analysis Research,* Technical Note 87, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (January 1974).

[Tenenbaum 74b] J.M. Tenenbaum, T.D. Garvey, S.A. Weyl, and H.C. Wolf, *ISIS: An Interactive Facility for Scene Analysis,* Technical Note 95, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (June 1974). Also published in *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition,* pp. 123–125, Copenhagen, Denmark (13–15 August 1974).

[Tomita 73] F. Tomita, M. Yachida, and S. Tsuji, "Detection of Homogeneous Regions by Structural Analysis," *Proc. 3rd Int. Jnt. Conf. on Artificial Intelligence,* Stanford, California, pp. 564–571 (20–23 August 1973).

[Tsuji 73] S. Tsuji and F. Tomita, "A Structural Analyzer for a Class of Textures," *Computer Graphics and Image Processing,* Vol. 2, No. 3/4, pp. 216–231 (December 1973).

[Wolfe 70] J.H. Wolfe, "Pattern Clustering by Multivariate Mixture Analysis," *Behavioral Research,* Vol. 5, pp. 329–350 (1970).