J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

IMAGE SEGMENTATION

An image for the purposes of this article is a two-dimensional array of intensities that we denote by $I(x, y)$, where x and y are discrete independent variables (the spatial variables) and $I(x, y)$ represents the intensity at a location (pixel) (x, y) and is quantized to be one of the admissible values. For example, $I(x, y) \in \{0, 1, 2, \ldots, 255\}$ is an image with 256 levels of gray. It is customary to think of gray level 0 as "black" and the gray level 255 as the color "white" and to think of the "origin" of the image as the top left corner. The image itself is merely an array of pixels, each of a certain intensity. There are, of course, images that have color information, or magnitude and phase information. This article primarily focuses on gray scale images. When images from these other sensors are available, it is often possible to reduce them to a gray scale format. The goal of image processing is to derive meaningful information from this two-dimensional arrangement of intensities.

In order to derive meaningful information from a two-dimensional arrangement of intensities, an image has to be processed. Often the first step in the processing of an image is segmentation. The goal of image segmentation is to detect objects that may be present in an image and to separate objects from each other and from the background. Due to its importance and its impact on subsequent processing, there has been a sustained contribution to the segmentation literature. Exactly which of these methods is most useful in a particular situation depends to a large extent on the nature of image itself. For example, if pixel intensities are locally uniform, then threshold or the edge-based segmentation techniques may be preferred. On the other hand, if the local pixel intensities vary but the pattern of variation is repeated, then texture-based segmentation may provide good results. If neither of these techniques provide good segmentation, then it may be possible to utilize a sequence of frames (object motion) to segment the image.

The aim of this article is to present some of the methods of image segmentation that individually, or in combination, can be used to segment a large proportion of images encountered in diverse applications.

Threshold-Based Image Segmentation

Thresholding is one of the easiest to apply of all the image segmentation schemes. In cases where the gray level intensity of an object is more or less uniform and greatly different from that of the background, thresholding also provides effective segmentation. In controlled situations, the efficacy of thresholding may be increased by choosing a background to have substantially different gray level characteristics than the object of interest.

Thresholding creates a thresholded image $T(x, y)$ from the original image $I(x, y)$ based on the choice of a threshold θ, where

$$T(x,y) = \begin{cases} 1 & \text{if } I(x,y) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

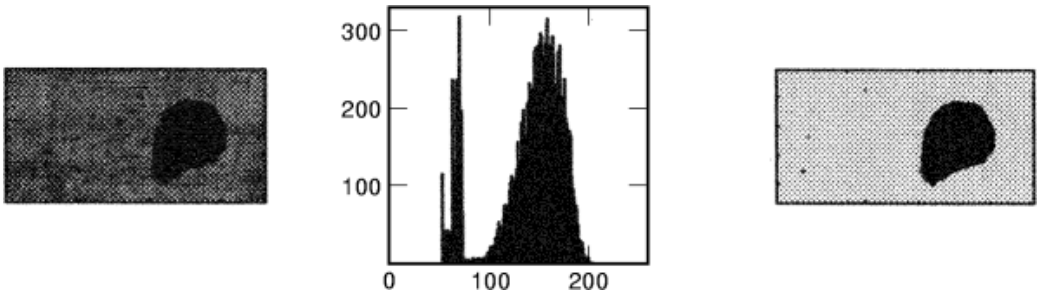

Fig. 1. A sample image of a wood board with a defect (left panel), the histogram of the image (middle panel), and the result of thresholding based on a threshold θ, which is set at the lowest point of the valley between the two peaks of the histogram.

If the threshold is appropriately chosen relative to the gray level of an object and the threshold, then $T(x, y)$ should be a monochrome image with the object being represented by a single value (either a 1 or a 0). Variations of the gray level can be removed so long as these variations are not large. Figure 1 shows a sample image of a wooden board that has a defect and its associated histogram. The board is clearly brighter than the defect. An appropriate choice for the threshold in such a case is the lowest point of the valley between the two peaks.

Machine-based estimation of the threshold thus has to rely on locating this lowest point between two peaks that (may) correspond to the object and the background. We present two methods for identifying the threshold.

The first method is based on finding the peaks in the histogram using a clustering algorithm such as the k-means algorithm. The k-means algorithm is based on starting with k randomly initialized cluster centers. Each input is assigned to the nearest cluster center, and after all inputs have been assigned, the cluster center locations are updated to be the centroid of the points assigned to the particular cluster center. By iterating the process, the cluster centers are distributed to occupy locations that have the highest density of inputs. Within the context of thresholding, the inputs are the gray levels of the pixels in the image, and for the segmentation of the image into two regions, $k = 2$. On convergence, the locations of the two cluster centers correspond to the most frequently occurring gray levels (the peaks of the histogram), and θ corresponds to the midpoint between the cluster center locations. If the gray level of each pixel is replaced by the index of the closest cluster center, then one obtains a two-tone segmented image.

The second method is based on parametric approximation (i.e., we approximate the shape of the histogram using a function of assumed form) (e.g., a polynomial of a certain degree). The minima can then be derived analytically based on the parameters of the function. For polynomials of high degree, there will be multiple candidates that may be ranked based on the characteristics of the threshold we seek. We cover approximation using parametric forms in greater detail in the section titled "Facet-Based Image Segmentation."

This idea of a single threshold can be extended to multiple thresholds, allowing multiple objects (each with substantially different gray levels) to be separated. Irrespective of the method used, the success of this global method of thresholding relies heavily on there being a bimodal (multimodal) histogram in which the modes are sufficiently separated by relatively deep valleys. To achieve such deep valleys, it is suggested in Ref. 1 that the histogram be computed only for those pixels where the Laplacian (second derivative of the change in intensity) is large.

There are further extensions, such as local thresholding, in which each disjoint square of the image is thresholded based on the histogram computed from pixel intensities in the square. This allows some degree of robustness against variations in illumination.

Edge-Based Image Segmentation

Edge-based image segmentation is based on the premise that $I(x, y)$ is locally smooth, and sudden transitions (or discontinuities) in $I(x, y)$ reflect the boundary between two distinct regions. For many images, specially those of synthetic objects, such is indeed the case. Pixel locations at which the intensity differs substantially from the pixel intensities of surrounding pixels are the *edge points*. The detection of these edge points forms the basis of edge-based image segmentation.

Mathematically, the rate of change of intensity along a given direction can be captured by the gradient. Consider a pixel (x, y). If we move along the x direction while holding y constant, then the change in intensity along the x direction can be written as

$$\nabla_x I(x,y) = \frac{\partial I(x,y)}{\partial x} \simeq \frac{I(x+h,y) - I(x,y)}{h} \tag{2}$$

Similarly, if we move along the y direction while holding x constant, then the change in intensity along the y direction can be written as

$$\nabla_y I(x,y) = \frac{\partial I(x,y)}{\partial y} \simeq \frac{I(x,y+h) - I(x,y)}{h} \tag{3}$$

Clearly, we could be moving in any direction (rather than in x or y; for example, northeast, southwest), and rather than computing the change in intensity in each of these directions, we can compute the gradient vector $\nabla I(x, y) = [\partial I(x, y)/\partial x \ \partial I(x, y)/\partial y]^T$, where T denotes the transpose. The directional derivative (rate of change of intensity) along any direction given by the vector $\boldsymbol{d}$ is then

$$\nabla_{\mathrm{d}} I(x,y) = \frac{\nabla I(x,y)^T \boldsymbol{d}}{\|d\|} = \nabla I(x,y) \cos\theta \tag{4}$$

where $\|\cdot\|$ is the magnitude of $\boldsymbol{d}$ and θ is the angle between $\nabla I(x, y)$ and $\boldsymbol{d}$. It is easy to see from Eq. (4) that $\nabla_d I(x, y)$ is maximum when $\boldsymbol{d}$ is in the same direction as $\nabla I(x, y)$. Consequently, the direction in which there is largest change in intensity is given by the gradient direction.

Thus the gradient magnitude $G_{\mathrm{m}}(x, y)$ at a pixel (x, y) is given by [as done here, we will often simply use ∇_x instead of the more cumbersome $\nabla_x I(x, y)$

$$G_{\mathrm{m}}(x,y) = \sqrt{\nabla_x^2 + \nabla_y^2} \tag{5}$$

Often, one can avoid the time-consuming operations of squares and square roots and approximate Eq. (5) as $|\nabla_x + \nabla_y|$. Similarly, the gradient direction, $G_{\mathrm{d}}(x, y)$, is

$$G_{\mathrm{d}}(x,y) = \tan^{-1}\left(\frac{\nabla_h}{\nabla_x}\right) \tag{6}$$

Equations (2) through (6) represent the basic equations on which a large class of edge detection operators are constructed. At each pixel we can simply compute the gradient magnitude using Eq. (5) and label pixels in the image with higher gradients as the edge pixels or the edge points. The manner in which ∇_x and ∇_y are computed leads to the different edge detection operators. In the following, we discuss many of these in detail.

Fig. 2. Roberts edge detector masks ∇_1 (left) and ∇_2 (right).

Roberts Cross Difference Operators. Roberts cross difference operators (2) are simple to implement and faster than most other types of edge detectors. Rather than finding the gradient along the x and y direction, Roberts operators are based on finding the gradient along two orthogonal diagonal directions. If we approximate the directional derivatives by forward difference, then ∇_1 and ∇_2, the masks for computing the gradient in the two diagonal directions, are given as shown in Fig. 2. Convolving the image with each of the masks gives $\nabla_1 I(x, y)$ and $\nabla_2 I(x, y)$.

The gradient magnitude and direction at (x, y) are then as given by Eq. (7):

$$G_{\mathrm{m}}(x, y) = \sqrt{\nabla_1^2 + \nabla_2^2} \qquad G_{\mathrm{d}}(x, y) = \tan^{-1}\left(\frac{\nabla_2}{\nabla_1}\right) - \frac{3\pi}{4} \tag{7}$$

Marking each pixel with a high gradient (higher than a threshold; say, the mean gradient magnitude over the image) with a 1 and those with low gradient a 0 results in a monochrome image that is nonzero at the edge points. Though fast and easy to compute, Roberts cross difference operators are sensitive to noise due to the small size of the mask that is used.

Sobel Operators. The Sobel operators (3) use 3×3 masks to compute ∇_x and ∇_y and are less sensitive to noise than Roberts operators. Part of the reason for reduced sensitivity to noise is that Sobel operators use a larger mask and, as we show, the Sobel operators are formed by averaging the directional derivatives.

Consider a pixel (x, y). Sobel operators are formed by estimating $\nabla_x I(x, y)$ and $\nabla_y I(x, y)$ using a weighted average of three symmetric differences; that is,

$$\begin{aligned}\nabla_x I(x, y) = \frac{1}{8}[&(I(x+1, y-1) - I(x-1, y-1)) \\ &+ 2(I(x+1, y) - I(x-1, y)) \\ &+ (I(x+1, y+1) - I(x-1, y+1))]\end{aligned} \tag{8}$$

and

$$\begin{aligned}\nabla_y I(x, y) = \frac{1}{8}[&(I(x-1, y+1) - I(x-1, y-1)) \\ &+ 2(I(x, y+1) - I(x, y-1)) \\ &+ (I(x+1, y+1) - I(x+1, y-1))]\end{aligned} \tag{9}$$

Note that $\nabla_x I(x, y)$ is estimated using a weighted average of symmetric differences as computed from the row above the pixel in question, the row in which the pixel is, and the row below the pixel in question [similar for $\nabla_y I(x, y)$]. It is because of this weighted averaging that the Sobel operators are more reliable edge detectors than Roberts edge detectors in the presence of noise. Masks corresponding to Eqs. (8) and (9) (we have dropped the constant multiplier since it does not make a difference in the computation of the edge points) used to compute ∇_x and ∇_y are shown in Fig. 3.

The gradient magnitude and direction can now be computed using Eqs. (5) and (6), and pixels with high gradient magnitude (higher than a threshold) are the edge points.

−1	0	1
−2	0	2
−1	0	1

−1	−2	−1
0	0	0
1	2	1

Fig. 3. Sobel edge detector masks ∇_x (left) and ∇_y (right).

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

0	1	0
1	−4	1
0	1	0

Fig. 4. 3×3 masks used for smoothing (left) and the Laplacian (right).

Laplacian of Gaussian. The edge points (i.e., locations where the gradient is large) are precisely the points where the second derivative (or the Laplacian) has a zero crossing. This can be easily seen from the fact that the gradient can be written as $\nabla I(x, y)$. The gradient is maximized when $\nabla(\nabla I(x, y)) = 0$. Consequently, $\nabla^2 I(x, y) = 0$ at the point where the gradient (rate of change of intensity) is maximized. However, due to the sensitivity to noise of the Laplacian, its use directly for detecting edge points is not recommended. Rather, the image is first smoothed by convolving it with a Gaussian $N(x, y)$, and then edge points are located based on the zero crossings of the Laplacian of the smoothed image (4). It is worth noting that smoothing by simple (unweighted) averaging over a local region of the image produces severe aliasing problems and is never recommended.

Digitally, the smoothing with a Gaussian and the Laplacian can be carried out by convolving the image with the masks shown in Fig. 4. Zero crossing at a pixel can be determined if the result of the convolution is smaller than $-z$ (larger than z) and of one of its eight neighbors is larger than z (smaller than $-z$). Zero crossings of the result of the convolution are then labeled as edge points.

Edge detection based on zero crossings of the Laplacian of Gaussian (LoG) filter is attractive since the smoothing operation allows for suppression of noise that may be present. While one may be tempted to use a Gaussian of large variance to obtain more pronounced smoothing, it comes at the expense of edge dislocation. Figure 5 shows a sample image and the edge points as obtained from the approaches presented previously. Note that because of the smoothing, many of the double edges that were close by and detected by Roberts and Sobel operators have combined into single edges in the LoG-based edge detection, demonstrating the trade-off between robustness and edge localization.

Difference of Gaussian. The LoG, $\nabla^2(N(x, y) * I(x, y))$, can be rewritten as $(\nabla^2 N(x, y)) * I(x, y)$ (i.e., we can take the Laplacian of Gaussian and convolve this result with the image; Ref. 4). Edges are marked where the result of the convolution has a zero crossing. Mathematically, the LoG can be expressed as (assuming independent normal variates along the x and y direction)

$$\begin{aligned} \nabla^2 N(x, y) &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= -\frac{1}{2\pi\sigma^4} \left(2 - \frac{x^2+y^2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{10}$$

where the variance in the x and y direction are equal and given by σ^2 (we want equal smoothing in any direction; hence the assumption of equal variance). Figure 6 shows the shape of $\nabla^2 N(x, y)$ for $\sigma = 0.4$.

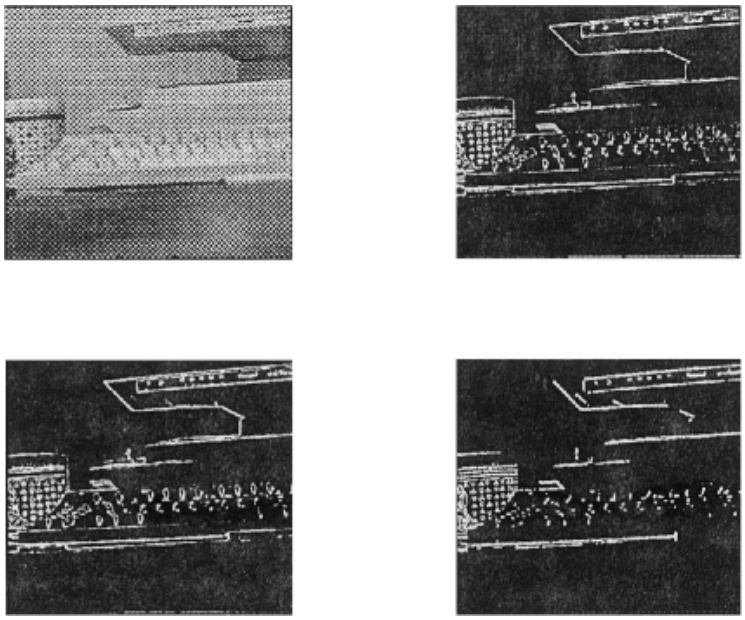

Fig. 5. A sample image (top left panel). The edge points as detected using Roberts cross-difference operators (top right panel), Sobel operators (bottom left panel), and zero crossing of the Laplacian of an image smoothed by a Gaussian (bottom right panel).

Fig. 6. Laplacian of Gaussian $\nabla^2 N(x, y)$.

The digital approximation of $\nabla^2 N(x, y)$ can in fact be done using the difference of two Gaussians (*DoG*), which have σ's approximately in the ratio 1.6:1 (4) [i.e., $\nabla^2 N(x, y) * I(x, y) \approx 4(N1(X, y) - N2(x, y)) * I(x, y)$, where $N1(x, y)$ and $N2(x, y)$ are the two Gaussians]. Figure 7 shows the digital masks for the two Gaussians, and Fig. 8 shows the LoG as realized using the DoG (compare with Fig. 6). As before, zero crossings of the DoG processed image are the edge points.

Facet-Model-Based Edge Detection. The facet model is based on the idea of parametric estimation. To clarify, consider a function of the form $I(x, y) = ax + by + c$. Here x and y are two independent variables, and a, b, and c are the parameters of the function. To make the connection with an image, $I(x, y)$ can be thought of as denoting the image intensity, and x and y are the spatial dimensions. At each pixel, then, we can adjust the parameters a, b, and c based on the intensity at the pixel and its neighbors, such that the function best fits (in a least squares sense) the local intensity profile. One such function can be defined at each pixel; all of these functions have the same form but the parameters a, b, and c are different. Such a model of an image

$\frac{1}{256}$

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

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Fig. 7. Masks corresponding to the smoothing by two Gaussians in the DoG-based segmentation.

Fig. 8. Digital realization of the Laplacian of Gaussian using a difference of Gaussian approach. The masks of Fig. 7 are used.

is piecewise linear. One can, of course, take more complex functions to model the local intensity profile of an image (quadratics and cubics are common) (5,6). In this sense, then, the image is made up of many facets (which, literally translated, means "many sides or faces").

It is worth noting that the more powerful the function (i.e., more the number of adjustable parameters), the more likely it is to fit any noise that may be present. To clarify this idea, Fig. 9 shows the intensities of one row of an image that has five columns. Though the facet model is applied to square neighborhoods, here we consider this one-dimensional simplification for ease of visualization. Shown in this figure are polynomial approximations of the intensity along this row using polynomials of degree 1, 2, and 4. Note that with a polynomial of degree 4 (i.e., with five adjustable parameters) it is possible for the function to pass exactly through each point.

Edge detection based on the facet model estimates the gradient directly in terms of the parameters of the function (i.e., based on a, b, and c in the preceding example and based on all the parameters if a more complex form for the function was assumed). For example, from $I(x, y) = ax + by + c$ one finds that

$$\nabla_x I(x, y) = a \qquad \text{and} \qquad \nabla_h I(x, y) = b \tag{11}$$

Thus, the gradient magnitude and direction are given by

$$\begin{aligned} G_{\mathrm{m}}(x, y) &= \sqrt{\nabla_x^2 + \nabla_y^2} = \sqrt{a^2 + b^2} \qquad \text{and} \\ G_{\mathrm{d}}(x, y) &= \tan^{-1}\left(\frac{\nabla_y}{\nabla_x}\right) = \tan^{-1}\left(\frac{b}{a}\right) \end{aligned} \tag{12}$$

From Eq. (12) it becomes clear that $\sqrt{a^2+b^2}$ gives the estimate of the gradient. Consequently, the center pixel in a small grid (which is used to obtain a, b, and c in the linear facet model) can be marked as an edge point if this expression evaluates to a relatively high value. Of course, to do this we need the values of the parameters a, b, and c.

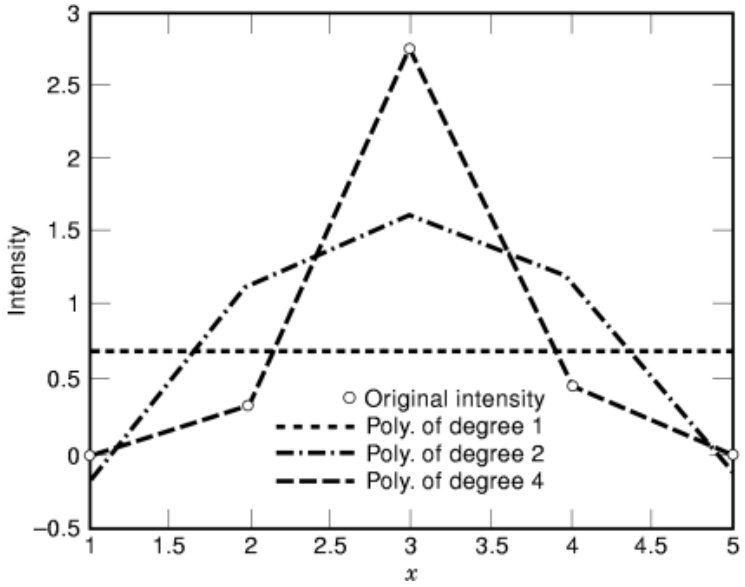

Fig. 9. Image intensity along a row and the best least squares sense approximation by polynomials of various degrees.

The estimation of the parameters of facet model can be done in a variety of ways. For example, consider the intensities in a 3×3 neighborhood R. Then the best fit (in a least squares sense) plane is one that satisfies as closely as possible the equation

$$I(x, y) = ax + by + c \qquad \forall x, y \in R \tag{13}$$

This leads to nine simultaneous equations, which in matrix form can be written as $AX = B$, where X is a 3×1 column vector of the parameters a, b, c; B is a 9×1 column vector whose elements are the intensities at each pixel in the 3×3 neighborhood; and A is a 9×3 matrix whose each row has the values of x, y, and 1 corresponding to the pixel whose intensity appears in the same row in B. The best least squares solution of the parameter vector $\boldsymbol{X}$ is given by the pseudoinverse, and we obtain the parameter vector as

$$\boldsymbol{X} = A^{+}B = (A^{T}A)^{-1}A^{T}B \tag{14}$$

Note that when the number of parameters are equal to the number of image points considered, Eq. (14) reduces to the standard inverse. This is consistent with the fact that in such a case, it is possible to find a function that passes through each point and the residual error is 0. In the preceding case, the pseudoinverse solution is the best solution in the least squared sense. We also note that Eq. (14) is a general formulation and is applicable irrespective of the particular parametric form that is assumed for the facet model. Quadratic and cubic facet models are thus easily accommodated, though the estimate of the gradient in these cases would change and be in terms of the parameters of those functions.

The availability of a function describing the intensity surface allows far greater flexibility than simple edge detection (for which the previously described methods may be sufficient). For example, the facet model framework can be used for noise removal from images (the residual error is assumed to be the noise).

A different approach along the line of function approximation is also possible (7). Here, rather than using a function with a fixed number of parameters, one can to find the lowest-order model (i.e., a function with the

least number of free parameters) that will approximate the function (image intensities) in a local patch of the image. The lowest-order model is sought under the constraint of producing a smooth approximation. To satisfy this constraint, one can successively increase the complexity of the function using a model selection criterion or use approximators that self-regulate the order (8). The order of the function required to approximate the local intensity surface can then be used to categorize local patches and produce a segmented image.

There are, of course, several other edge detectors. For example, Kirsch's (9) and Robinson's (10) compass masks each consist of eight masks; the Nevatia–Babu (11) compass masks consists of six masks. These methods are geared toward detecting edges in more than two directions. Often, the maximum response obtained from any of the mask is taken as the gradient magnitude. In Ref. 12, an edge detector is derived based on optimizing a detection and localization criteria. Canny's edge detector locates edges at points where the gradient of a Gaussian smoothed image has maximas. The definition of an edge itself can also be extended. For example, in Ref. 13, a pulse and staircase model of an edge is assumed. In Ref. 14, a covariance model of the edge is assumed. The choice of which detector to use depends on the characteristics of the edges under consideration. In situations where there are close neighboring edges, a staircase model of the edge is more appropriate.

Edge Linking

Once the set of edge points have been obtained using one of the methods described in the previous sections, we need to create several connected sequences of edge points. Each connected sequence of edge points then represents the object boundaries (or a part of the object boundary).

A simple method for edge linking is based on the concept of continuity (i.e., the gradient magnitude and direction should be comparable between two successive points). More formally, two *neighboring edge pixels* (x_1, y_1) and (x_2, y_2) can be connected if all of the following conditions are satisfied:

$$\begin{aligned} \|G_{\mathrm{m}}(x_1, y_1) - G_{\mathrm{m}}(x_2, y_2)\| &< a \\ \|G_{\mathrm{d}}(x_1, y_1) - G_{\mathrm{d}}(x_2, y_2)\| &< b \\ \|(x_2 - x_1)\| + \|(y_2 - y_2)\| &< c \end{aligned} \tag{15}$$

where a, b, c are small positive constants. It must, of course, be kept in mind that this approach, while sufficient for images acquired under controlled circumstances, may not work as well in the presence of noise, occlusion, or variable lighting conditions. It is worth noting that in some cases, such as in the detection of corners, we would like the gradient direction to change appreciably at the point we seek (i.e., the corner point). Condition 2 in Eq. (15) should be appropriately modified in such a case (see Ref. 15).

Hough Transform. The preceding heuristic for edge linking may not always produce good results. This is because it is trying to reconstruct a boundary based on local information and without knowing what the boundary looks like. A computationally efficient way of looking for boundaries of a simple parametric shape (such as a line or a circle) is based on the Hough transform (16). Extensions for the case of non-parametric shapes have also been proposed (17).

One way of conceptualizing the Hough transform is to find the number of edge points that satisfy the equation describing the shape with some specified shape parameters. For example, assume that the edge points detected in the image are at (x_1, y_1), (x_2, y_2), ..., (x_p, y_p) and we wish to find straight line boundary segments. Letting the equation of the boundary be defined by $y = mx + c$, where m is the slope and c the intercept, we can see how many of the edge points satisfy this equation for a given value of m and c. Corresponding to each choice of m and c (i.e., each different straight line), we obtain the number of edge points that lie on the line specified by m and c. Since m and c can take on a continuum of values, the Hough transform discretizes these parameters. For example, for the specific case of a line there are two parameters, and one can specify a

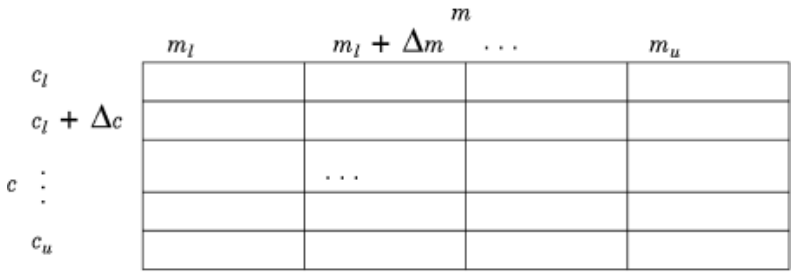

Fig. 10. The Hough accumulator array.

two-dimensional accumulator array, as shown in Fig. 10. In Fig. 10 the minimum and maximum value of the slope are shown as m_l and m_u; and Δm denotes the quantization of the slope. A similar notation is used for the intercept c. The content of each cell in the Hough accumulator array is the number of edge points that lie on the line whose parameters are some m and c.

Procedurally, the array is initialized to 0 and filled out as follows. Take an edge point, say (x_i, y_i), and obtain the corresponding value of $c_i \in \{c_l, cl + \Delta c, \ldots, c_u\}$ using the coordinates of the edge point and a given value of $m_i \in \{m_l, m_l + \Delta m, \ldots, m_u\}$. Increment the count at the accumulator corresponding to the intersection of m_i and c_i. Repeat for each allowed value of m_i and then for each edge point.

Once all edge points have been considered and the accumulator contents determined, the accumulators with the highest sums correspond to the line segments that are present. One can then link all the edge points that lie on a line specified by each cell with high accumulations. One final note: Since a vertical line has a slope of infinity, we cannot use the substitution method to obtain c_i corresponding to a m_i. Consequently, the polar (rather then the rectangular) form of a line is taken [i.e., $d = x \cos \theta + y \sin \theta$, where d is the perpendicular distance of the line from the (0, 0) point of the image]. The accumulator array is thus a quantization of d and θ, and one finds the value of θ corresponding to some value of d and the edge point (x_i, y_i) under consideration.

As mentioned before, the Hough transform can be used for any parametric or non-parametric shape. For example, if circles are to be detected, then the equation of the circle is $(x - x_c)^2 + (y - y_c)^2 = r^2$, where the center of the circle is (x_c, y_c) and its radius is r. The resulting accumulator array is three dimensional corresponding to the parameters that specify the circle.

Texture-Based Image Segmentation

While edge-based image segmentation relies on individual pixel intensity relative to intensity of the neighboring pixels, texture-based image segmentation relies on comparing the characteristics of a group of pixels with that of neighboring groups of pixels. Note that considering a group of pixels allows the usage of several variables to characterize the region (e.g., the particular arrangement of the pixel intensities within the group, the variance of the intensity, etc., as opposed to just the intensity when a single pixel is considered). The motivation for looking at a group of pixels arises from the fact that often individual regions in an image are not uniform in terms of *individual* pixel intensities but rather in terms of a *repetitive arrangement* of pixel intensities. Thus while the pixel intensities may vary arbitrarily within a small region, the pattern of variation is repeated. This periodicity of the arrangement of pixel intensities binds the region together and can be exploited in the segmentation of images.

Intuitively, we term a repetitive arrangement of pixel intensity as texture. We also associate several (subjective) descriptors with texture (e.g., fine grained, coarse grained, strongly ordered, weakly ordered). To

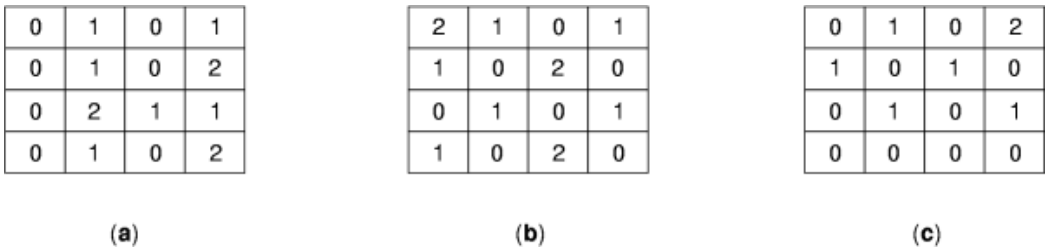

Fig. 11. Simple images used to illustrate the co-occurrence matrix based characterization of texture.

exploit texture for machine-based image segmentation, we focus on two approaches that provide quantitative descriptors for texture. These approaches are based on (1) the co-occurrence matrix and (2) the fractal dimension. Though we restrict discussion to these two approaches, several other approaches based on autoregression, spectral, and Markov random field models have also been proposed (for a discussion, see Ref. 32). We choose these two approaches because of space limitation and because between the two it allows us to account for most textures (while the first approach is more appropriate for oriented textures—it requires a direction to be specified—the second is more appropriate for disoriented textures). Furthermore, many studies have indicated that the results provided by the co-occurrence matrix formulation are superior to those obtained from spectral-based approaches. We describe each of them and indicate their use in image segmentation.

Co-Occurrence Matrix. Perhaps the most common of the statistical approaches to texture characterization is based on the co-occurrence matrix. The co-occurrence matrix, C, is a symmetric $l \times l$ matrix, where l is the number of gray levels in the image. An element C_{ij} of this matrix reflects the joint probability of two pixels co-occurring relative to each other with gray levels g_i and g_j. The relative position must be chosen so as to find the suspected placement of intensities (texture). Examples of the relative position are along a specific diagonal separated by a distance of 1, or the four nearest neighbors, and so on.

Consider, for example, the images shown in Fig. 11, in which the number of gray levels $l = 3$. In each of these images there is a vertical and/or diagonal band, even though these are not the only structures that are present. For example, in Fig. 11(c) there is a specific pattern formed by the four neighbors around pixels with locations (2, 2) and (3, 3). However, since there is vertical banding in each of the three images, let us specify the relative position as *along a vertical line separated by one pixel*. C_{ij} would thus reflect the joint probability of finding gray levels g_i and g_j separated by one pixel along the vertical.

The co-occurrence matrices for the three images in Fig. 11 are shown in Fig. 12. C_{00}, for example, in Fig. 12(a) shows the total number of locations where gray level 0 was found one pixel away in the vertical direction from gray level 0 (as specified by the relative position). To be a true probability, each of the elements in the co-occurrence matrix should be divided by all pairs of pixels [i.e., $(2)_{mn}$, where m and n are the number of rows and columns in the image]. This divisor has been omitted for clarity. Note that the co-occurrence matrix in Fig. 12(a) shows large elements at C_{00}, implying that 00 occurs fairly often in the vertical direction, whereas in Figs. 12(b) and 12(c) C_{01} (or C_{10}) is large, implying frequent co-occurrence of these gray levels in a vertical band direction separated of course by one pixel.

$$\begin{bmatrix} 8 & 2 & 0 \\ 2 & 2 & 5 \\ 0 & 5 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 8 & 3 \\ 8 & 0 & 1 \\ 3 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 4 & 9 & 1 \\ 9 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

(a) (b) (c)

Fig. 12. Co-occurrence matrices for the three images shown in Fig. 11, assuming the position operator $\boldsymbol{d}$ to be one pixel down. The divisor for each element $\Sigma_{ij}C_{ij}$ is not shown.

The characterization of the co-occurrence matrix can be done based on several statistical features that are computed from it. Commonly extracted features are defined as follows:

$$\begin{aligned} \text{Entropy} &= -\sum_{ij} C_{ij} \log C_{ij} \\ \text{Uniformity} &= \sum_{ij} C_{ij}^2 \\ \text{Moment of order } k &= \sum_{ij} |i-j|^k C_{ij} \\ \text{Compactness} &= \sum_{ij} (i+j-2\mu) C_{ij} \end{aligned} \tag{16}$$

Indeed other features can be computed from the co-occurrence matrix (such as correlation and inverse moment; Ref. 18). Each of these features reflects the state of the co-occurrence matrix. For example, the entropy is maximized when all elements of the matrix are equal; the moment is small for a predominantly diagonal C.

From a practical viewpoint, it is best to reduce the number of gray levels in an image prior to computing the co-occurrence matrix. Though this leads to computational simplicity (the matrix has l^2 elements), it also makes the occurrence of a very sparse matrix less likely. Further, one may consider using a few relative position operators and extracting the aforementioned features from each of the resulting co-occurrence matrices and treating, for example, the entropy from each case to be a unique feature.

From an image segmentation viewpoint, the goal is to segment regions with different texture based on the features extracted from the gray level co-occurrence matrix. Conceptually, the process is as follows. The original image is decomposed into disjoint squares (say 32×32). Features are extracted from each of these squares and adjoining regions merged if the extracted features are similar. If a disjoint square is not merged, then it is further decomposed into smaller squares (say four 16×16 squares). Features from these smaller squares are computed, and they are either merged with the larger squares or further decomposed. This split and merge technique can be used to segment an image into regions that are uniform in texture.

Fractal Dimension. Fractals may be variously defined, but the most intuitive definition is that a fractal is a shape made of parts similar to the whole (19) (imagine a cauliflower or the image of a concrete wall). It is also worth noting that not all self-similar objects are fractals (for example, a line, square, or a cube are not fractal).

Mathematically, the concept of a (fractal) dimension can be defined as

$$D = \frac{\log p}{\log \lambda} \qquad \text{or} \qquad p = \lambda^D \tag{17}$$

where p is the number of self-similar pieces and λ is the scaling (or the magnification) factor. For example, a square can be divided into N^2 equal disjoint squares are each of whose side is $1/N$ of the original square's side; consequently the magnification factor is N. From Eq. (17), the dimension of the square is thus $\log N^2/\log N = 2$. On the other hand (this is the famed Sierpinski gasket), one can form three self-similar equilateral triangles from a given equilateral triangle. The sides of the resulting self-similar triangles are 1/2 of the original sides. The fractal dimension of this object is thus $\log 3/\log 2 = 1.58$.

Interestingly, it has also been shown that images of fractal surfaces are also fractal (20). To clarify, one may note from Eq. (17) that a change in the scaling factor by an amount z causes a scaling in the number of self-similar pieces p by a factor of z^D. In images of fractal surfaces, a scaling by z in x and y causes scaling of z^H in the intensity $I(x, y)$, where $H = 3 - D$, or

$$E[(I(x_1, y_1) - I(x_2, y_2))^2] = ((x_2 - x_1)^2 + (y_2 - y_1)^2)^H \qquad (18)$$

Equation (18) provides a direct way of estimating H (or, equivalently, D). One simply finds the best fit line in the log-log space of intensity difference and displacement. The segmentation of an image can thus be achieved in a manner similar. Here one can estimate the fractal dimension D for small disjoint squares and use the split and merge procedure outlined previously to segment the image.

Region Growing

The methods of segmentation discussed in the previous section were based on a single attribute (e.g., intensity based or texture based). Region growing, on the other hand, provides a framework that allows for the simultaneous use of multiple attributes that characterize a pixel. Let the features characterizing a pixel located at (x, y) be denoted by the column vector $\boldsymbol{Z}(x, y)$. A region in this context denotes a set of pixels, and the process begins with a single region $R^{(i)}$ that has a single arbitrarily chosen pixel in it (a better way to choose this initial seed point can be based on the distance transform; see the next section). Let the vector $\bar{\boldsymbol{Z}}^{(i)}$ denote the average of the feature vectors in $R^{(i)}$. Then a new pixel, which is the neighborhood of the pixels in $R^{(i)}$, is considered for addition to $R^{(i)}$ if

$$\|Z(x, y) - \bar{Z}^{(i)}\| < \epsilon \qquad (19)$$

where ε is a small tolerance. Having a small value for ε promotes the growth of a region whose feature variance is small. The distance between the two vectors in Eq. (19) can be the L_2 norm (distance in the Euclidean sense), the L_∞ norm defined as $\max_j |Z_j(x, y) - \bar{Z}^{(i)}{}_j|$, where the subscript j indexes the individual features in the respective feature vector, or a weighted distance with weights selected to give more dominance to certain features in the feature vector. If Eq. (19) is satisfied by the pixel under test, then one includes it in the region and precludes it from being considered in another growing operation. When no pixel in the neighborhood of the pixels in $R^{(i)}$ can be added to $R^{(i)}$, one starts a new region. The process is continued until each pixel in the image has been added to a region.

Distance-Transform-Based Segmentation

Distance-transform-based segmentation is most useful in separating two overlapping objects that are similar based on other characteristics (such as average intensity and texture). The essence of distance-transform-based segmentation is to estimate the distance of each pixel from a nearest background pixel and then to use the

Fig. 13. A sample image (left panel) thresholded to create a binary image (middle panel), and the resulting distance transform (right panel).

watershed algorithm to split the objects. Exactly how the distance to the background is measured (Euclidean, city-block distance, etc.) gives rise to variations of the distance transform.

To compute the distance transform, it is necessary to obtain a binary image $B(x, y)$ of the original image $I(x, y)$. The binary image could be produced using thresholding, as discussed previously. There exists an efficient procedure due to Ref. 21 that computes the distance transform in two passes based on the city-block distance. The first of these passes moves from the top left corner of the image toward the bottom right corner, and the other from the bottom right corner of the image to the top left corner. In this traversal, the distance to a background pixel can be accumulated. Though edge effects (i.e., a single-pixel-wide border around the image) can be accomodated, an easier to apply procedure (and one almost as accurate) is based on creating two scratch variables, $G1$ and G, both initialized to 0's and having the same size as the original image and restricting the computation to within one pixel of the image boundary.

The first pass (top left corner to the bottom right corner) then has

$$G1(x, y) = B(x, y)(1 + \min\{G1(x - 1, y), G1(x, y - 1)\}) \quad (20)$$

The second pass (bottom right corner to the top left corner) then has

$$G(x, y) = \min\{G1(x, y), (G(x + 1, y) + 1), (G(x, y + 1) + 1)\} \quad (21)$$

$G(x, y)$ is the resultant distance transform.

Figure 13 shows the distance transform obtained for a sample image. Note that brighter intensities indicate larger distances from the background. The watershed algorithm to separate the overlapping object is then performed as follows. Starting with the largest possible threshold, one progressively decreases the threshold until as many points as the number of overlapping objects are nonzero. These correspond to the peaks in G (i.e., locations that are furthest from the boundary). If the object sizes are not comparable, then one obtains progressively more nonzero points and then a disjoint point corresponding to the interior of the other object. The threshold is further lowered until disjoint areas start touching each other. This appearance of the peaks is, of course, a function of the object shape (convexity, etc.) and the relative sizes of the objects. Depending on the application, heuristics can be adopted to separate the objects. Region growing can be further used to refine the object boundary.

Motion-Based Image Segmentation

All the image segmentation methods discussed in the previous sections are based on a single image frame. Motion, however, provides valuable cues that can aid in the segmentation, especially in situations where intensity or texture do not provide enough information to partition the image into meaningful regions.

We first consider the effect of motion in the frequency domain (23,24,25). Consider an image denoted by $I(x, y, t)$ where t indexes the particular frame. To simplify, let us assume that there is no motion in the y direction. The image can thus be represented in reduced form as $I(x, t)$. The two-dimensional Fourier transform of $I(x, t)$ is given by

$$F(u, w) = \iint_{-\infty}^{\infty} f(x, t) e^{-j2\pi(ux+wt)}\, dx\, dt \tag{22}$$

If we now consider a displacement of d along the x direction, then the Fourier transform of this new image $f(x - d, t)$ is

$$\begin{aligned} \mathcal{F}\{f(x-d,t)\} &= \iint_{-\infty}^{\infty} f(x-d,t) e^{-j2\pi(ux+wt)}\, dx\, dt \\ &= e^{-j2\pi ud}\left[\iint_{-\infty}^{\infty} f(z,t) e^{-j2\pi(uz+wt)}\, dz\, dt\right] \end{aligned} \tag{23}$$

where $z = x - d$. Equation (23) can easily be seen to be

$$\mathcal{F}\{f(x-d,t)\} = e^{-j2\pi ud} F(u, w) \tag{24}$$

Thus a stationary image has its spectrum along the u axis. With translation, the spectrum is sheared along the complex (w) axis. It is trivial to extend this when the motion is both in the x and y directions. It therefore becomes possible to detect motion and consequently identify and segment the object.

It is also possible to establish motion based on the estimation of feature densities (26). The general idea followed is to obtain a non-parametric estimation of the feature density in the first frame, such that the asymptotic local density of some "control points" approaches the local density of the features. If these control points are now allowed to move to estimate the density of features obtained in the second frame, then under the constraint of topology preservation, the movement of the control points establishes correspondence in the two frames and allows one to recover the movement of points. If different objects in an image exhibit movement in different directions, one can segment the objects based on areas of homogeneous movement (assuming rigidity of the object). The features extracted in Ref. 26 were simply edge points. It is also possible to follow alternate matching schemes based on higher-level features such as lines, edges, vertices, and surfaces (27).

Scale Space and Image Segmentation

The notion of scale is fundamental to the processing of images (and signals in general). Assume that from a given image, we produce a sequence of images. Each image in the sequence is obtained by smoothing the original image by an increasing amount. Smoothing by larger amounts then corresponds to a convolution of the image with Gaussians of increasing variances. We encountered a form of this in the DoG approach to edge detection, in which the image was smoothed by Gaussians of two different variances. In an image resulting from smoothing with a Gaussian of small variance (the finer scale), much of the detail is present in the resultant

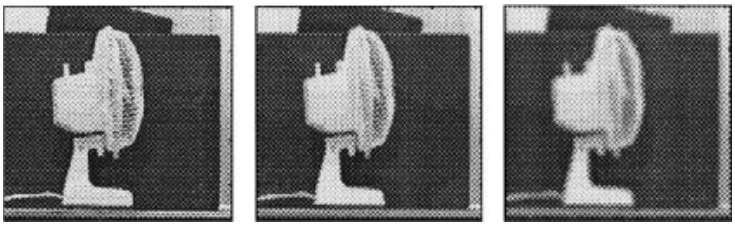

Fig. 14. A sample image (left panel) and images produced by smoothing with Gaussians of increasing variances.

Fig. 15. Edges detected for each of the images shown in Fig. 14.

image. In an image resulting from smoothing with a Gaussian of large variance (the coarse scale), much of the gray level variations have been removed and the image appears blurred and out of focus (see Fig. 14). From this perspective the scale is defined in terms of the *scale parameter*—the variance of the Gaussian used for smoothing. The original image and the sequence of images obtained with increasing amounts of smoothing together constitute the *scale space* of the image. It is also worth noting that the significant edges that are present in the image also *persist* over a much larger interval of the scale parameter. For example, all the panels in Fig. 14 show the persistence of the edges that would normally be associated with the image.

Of course, it is true that the image at the finest scale (the original image) has all the information that any image at a coarser scale has. However, the utility of the scale-space representation is based on the fact that real images may contain noise (which may be noise from the camera or any other gray level variation that is not of importance) and the fact that the objects themselves in an image are of different sizes. To make the processing more robust, we might consider a larger mask than the usual 3×3 mask that we used to compute the edges before. However, if the mask is made too large, we might miss some of the structures that are present in the image. If the mask is small, then we may be more sensitive to noise. Consequently, there is a scale at which an object is more easily detected. Since an image may have many objects, each with its own scale, there needs to be a scale-space representation of the image.

For the scale-space representation to be useful, it must possess some properties. The most important is that as one moves from a finer scale to a coarser scale, structures (features or gray level patterns in the image) must disappear monotonically and that no new structures are introduced. Second is the concept of invariance, which implies that the structures discovered should be independent of monotonic transformation of the intensity, or under rescaling of the spatial axis (i.e., $x \rightarrow sx$ and $y \rightarrow sy$). Third, for obvious reasons we require isotropy and homogeneity. It turns out that the Gaussian operator used previously uniquely satisfies these conditions and is thus used for generating a scale space.

The segmentation of an image from a scale-space perspective thus utilizes the tools we have discussed in previous sections but couples it with the notion of persistence. That is, detections (be they edges or regions) are considered significant if they appear for a wide range of the scale parameter. For example, in the segmentation of Fig. 14 shown in Fig. 15, the outline of the fan (the more significant edge) persists over the entire scale interval; however, the grill surrounding the fan does not.

Some final notes. First, it may also be observed that the information present in coarser levels of the scale space is significantly less (than at finer levels). Consequently, it is possible to subsample the image at coarse resolutions and reduce the size at these levels. One thus ends up with a pyramidal structure in which the base of the pyramid constitutes the image at the finest scale and the higher levels of the pyramid constitute the coarser levels. The size of the image as one travels up usually reduces by a factor of 2 in each dimension (i.e., if the base of the pyramid is $2^N \times 2^N$, then the next higher level is $2^{N-1} \times 2^{N-1}$). This reduces the storage required.

Second, deriving a scale-space description of an image based on Gaussian smoothing corresponds to embedding the signal in a one-parameter family of derived signals. The one parameter refers to σ, which is increased to produce more pronounced smoothing. It is also possible to create a multiscale representation based on embedding a signal in a two-parameter family of derived signals (31). Here the two parameters correspond to the translation and dilation of a single function, often called the "mother" wavelet and shown as follows:

$$\psi_{a,b}(x) = |a|^{-p}\psi\left(\frac{x-b}{a}\right) \qquad a \neq 0, p \geq 0 \qquad (25)$$

where the parameters $a$and b correspond to the dilation and translation parameters, respectively. The scale-space analysis as presented previously is thus contained within this more general wavelet-based description.

Conclusion and Future Directions

Image segmentation is one of the first steps in the automated processing of images. Consequently, it has benefited from at least three decades of research activity. Often, many approaches are unique to the particular type of image being processed (for additional details, Refs. 32,33,34,35,36 are excellent). The topics presented in this article were chosen since they have applicability across a broad range of images. The broad applicability notwithstanding, any practical implementation must choose a subset of these techniques based on the content of the image. Some topics such as morphological image processing should also be considered; they were not presented here due to the defined scope of the article.

Despite the significant contributions to the area of image segmentation, it is likely that the increased storage and computational ability of the present and the near future will usher in more sophisticated methods of segmentation. For example, most methods (such as the masks presented in this article) are based on assumed signal and noise characteristics. Their success in a particular situation is a function of the validity of those assumptions. As a corollary, adaptive methods of segmentation are likely to lead to improved segmentation methods and additionally provide a framework in segmentation that can be interlaced with understanding. Based on the incremental understanding obtained from partial segmentation, it is possible to adapt the segmentation strategy to be consistent with the objects and their scale in the image.

The current paradigm that underlies much of image processing is based on reducing the amount of data monotonically through the process. While this allows the development of algorithms within the confines of present (or past) computational ability, it also deprives us of the significant benefits that *increasing* the dimensionality of the input makes possible. Such an increase of dimensionality often simplifies a problem considerably. The use of motion is but one example. A random pattern against a random background is difficult to detect in a static image if the mean of the pattern is comparable to the mean of the background. When the dimensionality of the input is increased through the inclusion of motion, a moving random pattern immediately becomes visible. Such an increase in dimensionality can also be synthesized based on the higher-order relationships of pixel intensities. Indeed, it is surprising (or perhaps not so) that the neuron population in our own visual cortex is at least two to three orders of magnitude more than the number of neurons in the optic nerve.

BIBLIOGRAPHY

1. J. S. Weszka R. N. Nagel A. Rosenfeld A threshold selection technique, *IEEE Trans. Comput.*, **C-23**: 1322–1326, 1974.
2. L. G. Roberts Machine perception of three dimensional solids, in J. D. Tippet et al. (eds.), *Optical and Electro-Optical Information Processing*, Cambridge, MA: MIT Press, 1965, pp. 159–197.
3. I. E. Sobel *Camera models and machine perception*, Ph.D. Dissertation, Dept. Electrical Engineering, Stanford Univ., 1970.
4. D. Marr E. Hildreth Theory of edge detection, *Proc. R. Soc. London*, **B207**: 186–217, 1980.
5. R. M. Haralick L. T. Watson A facet model for image data, *Comput. Graph. Image Process.* **15**: 113–129, 1981.
6. R. M. Haralick Cubic facet model edge detector and ridge valley detector: Implementation details, in E. S. Gelsema and L. Kanal (eds.), *Pattern Recognition in Practise II*, Amsterdam, The Netherlands: North Holland, 1986.
7. R. Kothari D. Ensley On the use of model order for detecting potential target locations in SAR images, *Proc. SPIE Algorithms Synthetic Aperture Radar Imagery V*, **3370**: 1998.
8. K. Agyepong R. Kothari Controlling hidden layer capacity through lateral connections, *Neural Computat.*, **9** (6): 1381–1402, 1997.
9. R. Kirsch Computer determination of the constituent structure of biologic images, *Comput. Biomed. Res.*, **4**: 315–328, 1971.
10. G. S. Robinson Edge detection by compass gradient masks, *Comput. Graph. Image Process.*, **6**: 492–501, 1977.
11. R. Nevatia K. R. Babu Linear feature extraction and description, *Comput. Graph. Image Process.*, **13**: 257–269, 1980.
12. J. Canny A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, **8**: 679–698, 1986.
13. M. Shah A. Sood Pulse and staircase edge models, *Comput. Vision Graph. Image Process.*, **34**: 321–343, 1986.
14. F. Heijden Edge and line feature extraction based on covariance models, *IEEE Trans. Pattern Anal. Image Process.*, **17**: 16–33, 1995.
15. R. Deriche G. Giraudon Accurate corner detection: An analytical study, *Proc. Int. Conf. Comput. Vision*, 1990, pp. 66–70.
16. P. V. C. Hough *Methods and means for recognizing complex patterns*, US Patent 3,069,654, 1962.
17. D. H. Ballard Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recog.*, **13**: 111–122, 1981.
18. T. Pavlidis *Structural Pattern Recognition*, New York: Springer-Verlag, 1977.
19. B. B. Mandelbrot Self-affine fractal sets, in L. Pietronero and E. Tosatti (eds.), *Fractals in Physics*, Amsterdam, The Netherlands: North Holland, 1986.
20. P. Kube A. Pentland On the imaging of fractal surfaces, *IEEE Trans. Pattern Anal. Mach. Intell.*, **10**: 704–707, 1988.
21. A. Rosenfeld J. Pfaltz Distance functions in digital pictures, *Pattern Recog.*, **1**: 33–61, 1968.
22. G. Borgefors Distance transformations in digital images, *Comput. Vision Graph. Image Process.*, **34**: 344–371, 1986.
23. E. H. Adelson J. A. Movshon Phenomenal coherence of moving visual patterns, *Nature*, **300**: 523–525,1982.
24. A. B. Watson A. J. Ahmuda, Jr. Model of human visual-motion sensing, *J. Opt. Soc. Amer. A*, **2**: 322–341, 1985.
25. J. P. H. van Santen G. Sperling Elaborated Reichardt detectors, *J. Opt. Soc. Amer. A: Opt. Image Sci.*, **2**: 300–321, 1985.
26. R. Kothari J. Bellando Optical flow determination using topology preserving mappings, *Proc. IEEE Int. Conf. Image Process.*, 1997, pp. 344–347.
27. V. Venkateswar R. Chellapa Hierarchical stereo and motion correspondence using feature groupings, *Int. J. Comput. Vis.*, **15**: 245–269, 1995.
28. A. P. Witkin Scale-space filtering, *Proc. Int. Joint Conf. Artificial Intell.*, 1983, pp. 1019–1022.
29. J. J. Koenderink The structure of images, *Biological Cybern.*, **53**: 383–396, 1986.
30. T. Lindeberg *Scale-Space Theory in Computer Vision*, Norwell, MA: Kluwer, 1994.
31. I. Daubechies Orthonormal bases of compactly supported wavelet, *Comm. Pure Appl. Math.*, **XLI**: 167–187, 1987.
32. R. M. Haralick L. G. Shapiro *Computer and Robot Vision*, vol. I, Reading, MA: Addison-Wesley, 1992.
33. R. Gonzalez R. Woods *Digital Image Processing*, Reading, MA: Addison-Wesley, 1992.
34. A. Jain *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
35. A. Rosenfeld A. C. Kak *Digital Picture Processing*, New York: Academic Press, 1982.
36. K. R. Castleman *Digital Image Processing*, Upper Saddle River, NJ: Prentice-Hall, 1996.

RAVI KOTHARI
University of Cincinnati