Computer Vision

Course 10

Edge Linking and Boundary Detection

Ideally, edge detection should yield sets of pixels lying only on edges. In practice, these pixels seldom characterize edge completely because of noise, breaks in the edges due to nonuniform illumination, and other effects that introduce fake discontinuities in intensity values. Therefore, edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries. We discuss three fundamental approaches to edge linking that are representative of techniques used in practice. The first requires knowledge about edge points in a local region; the second requires that points on the boundary of a region be known; the third is a global approach that works with an entire edge image.

Local processing

A simple way to link edge points is to analyze the characteristics of pixels in small neighborhoods about every point ***(x, y)*** that has been declared an edge point. All points that are similar according to predefined criteria are linked, forming an edge of pixels that share common properties according to the specified criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are:

(1) the strength (magnitude)

(2) the direction

of the gradient vector. Let $\boldsymbol{S_{xy}}$ denote the set of coordinates of a neighborhood centered at $\boldsymbol{(x, y)}$ in an image. An edge pixel with coordinates ***(s, t)*** in $\boldsymbol{S_{xy}}$ is similar in *magnitude* to the pixel at ***(x, y)*** if:

$$|M(s,t)-M(x,y)|\leq E\;,\quad E>0$$ - positive threshold.

An edge pixel with coordinates ***(s, t)*** in $\boldsymbol{S_{xy}}$ has an *angle* similar to the pixel at ***(x, y)*** if:

$$|\alpha(s,t)-\alpha(x,y)|\leq A\;,\quad A>0$$ - positive angle threshold.

The direction of the edge at ***(x, y)*** is *perpendicular* to the direction of the gradient vector at that point.

A pixel with coordinates ***(s, t)*** in $\boldsymbol{S_{xy}}$ is linked to the pixel at ***(x, y)*** if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple procedure would be to assign a different intensity value to each set of linked edge pixels.

The preceding formulation is computationally expensive because all neighbors of every point have to be examined. A simplification particularly well suited for real time applications consists of the following steps:

1. Compute the gradient magnitude and the angle arrays $M(x, y)$and $\alpha(x, y)$ of the input image $f(x, y)$.
2. Form a binary image g:

$$g(x,y)=\begin{cases}1 & \text{if } M(x,y)>T_M \ AND\ \alpha(x,y)=A\pm T_A \\ 0 & \text{otherwise}\end{cases}$$

where T_M is a threshold, A is a specified angle direction, and $\pm T_A$ defines a "band" of acceptable directions about A

3. Scan the rows of $\boldsymbol{g}$ and fill (set to $\boldsymbol{1}$) all gaps (sets of $\boldsymbol{0}$s) in each row that do not exceed a specified length $\boldsymbol{K}$. Note that a gap is bounded at both ends by one or more $\boldsymbol{1}$s. The rows are processed individually, with no memory between them.

4. To detect gaps in any other direction $\boldsymbol{\theta}$, rotate $\boldsymbol{g}$ by this angle and apply the horizontal scanning procedure in Step 3. Rotate the result back by $-\boldsymbol{\theta}$.

In general, image rotation is an expensive computational process so, when linking in numerous angle directions is required, it is more practical to combine Steps 3 and 4 into a single radial scanning procedure.

Figure 10.27(a) shows an image of the rear of a vehicle. The objective of this example is to illustrate the use of the preceding algorithm for finding rectangles whose sizes makes them suitable candidates for license plates. The formation of these rectangles can be accomplished by detecting strong horizontal and vertical edges.

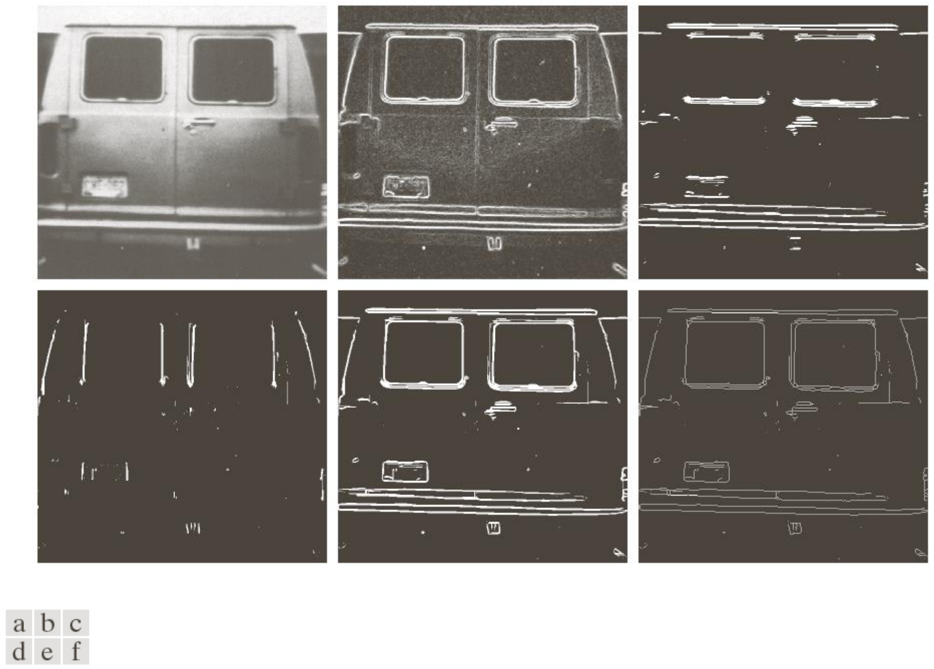


FIGURE 10.27 (a) A 534 × 566 image of the rear of a vehicle. (b) Gradient magnitude image. (c) Horizontally connected edge pixels. (d) Vertically connected edge pixels. (e) The logical OR of the two preceding images. (f) Final result obtained using morphological thinning. (Original image courtesy of Perceptics Corporation.)

T_M =30% of the maximum gradient value, $A=90^o$, $T_A = 45^o$, $K=25$.

Regional processing

Often the location of the regions of interest in an image is known or can be determined. This implies that knowledge is available regarding the regional membership of pixels in the corresponding edge image. We can use techniques for linking pixels on a regional basis, with the desired result being an approximation to the boundary of the region. One approach is to fit a 2-D curve to the known points. Interest lies in fast-executing techniques that yield an approximation to essential features of the boundary, such as extreme points and concavities. Polygonal approximations are particularly attractive because they capture the essential shape features of a region while keeping the representation of the boundary relatively simple. We present an algorithm suitable for this purpose.

Two important requirements are necessary. First, two starting points must be specified; second, all the points must be ordered (e.g. in a clockwise or counter clockwise direction).

An algorithm for finding a polygonal fit to open and closed curves may be stated as follows:

1. Let ***P*** be a sequence of ordered, distinct, ***1***-valued points of a binary image. Specify two starting points, ***A*** and ***B***. These are the two starting vertices of the polygon.

2. Specify a threshold ***T***, and two empty stacks OPEN and CLOSED.

3. If the points in ***P*** correspond to a closed curve, put ***A*** into OPEN and put ***B*** into OPEN *and* into CLOSED. If the points correspond to an open curve, put ***A*** into OPEN and put ***B*** into CLOSED.

4. Compute the parameters of the line passing from the last vertex in CLOSED to the last vertex in OPEN.

5. Compute the distance from the line in Step 4 to all points in ***P*** whose sequence places them between the

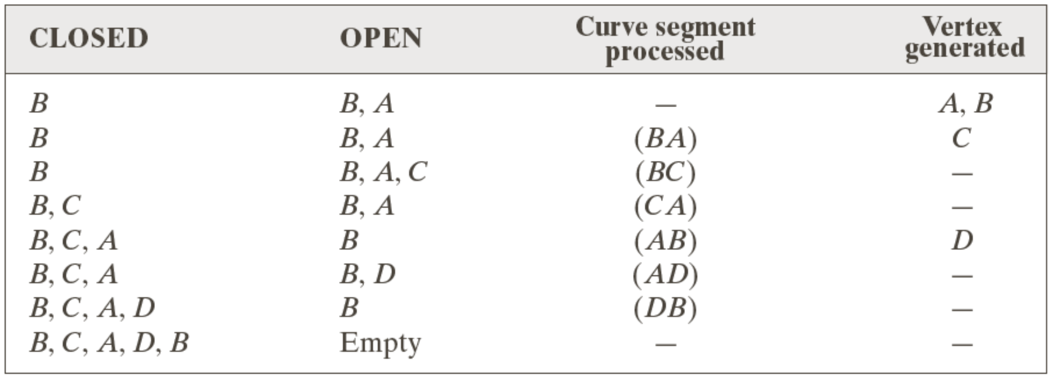
vertices from Step 4. Select the point $\boldsymbol{V_{max}}$ with the maximum distance $\boldsymbol{D_{max}}$ (ties are resolved arbitrarily)

6. If $\boldsymbol{D_{max} > T}$, place $\boldsymbol{V_{max}}$ at the end of the OPEN stack as a new vertex. Go to step 4.

7. Else, remove the last vertex from OPEN and insert it as the last vertex in CLOSED.

8. If OPEN is not empty go to Step 4.

9. Else, exit. The vertices in CLOSED are the vertices of the polygonal fit to the points in ***P***.



CLOSED	OPEN	Curve segment processed	Vertex generated
B	B, A	—	A, B
B	B, A	(BA)	C
B	B, A, C	(BC)	—
B, C	B, A	(CA)	—
B, C, A	B	(AB)	D
B, C, A	B, D	(AD)	—
B, C, A, D	B	(DB)	—
B, C, A, D, B	Empty	—	—

TABLE 10.1
Step-by-step details of the mechanics in Example 10.11.

Course 10

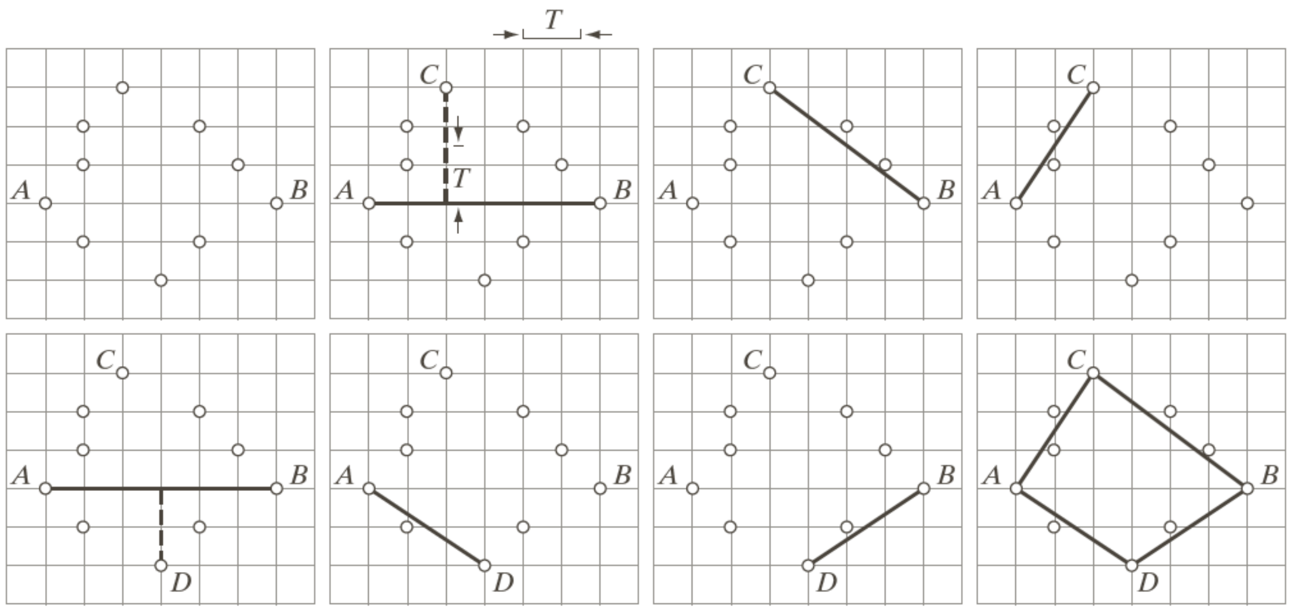


FIGURE 10.29 (a) A set of points in a clockwise path (the points labeled *A* and *B* were chosen as the starting vertices). (b) The distance from point *C* to the line passing through *A* and *B* is the largest of all the points between *A* and *B* and also passed the threshold test, so *C* is a new vertex. (d)–(g) Various stages of the algorithm. (h) The final vertices, shown connected with straight lines to form a polygon. Table 10.1 shows step-by-step details.

Global processing using the Hough transform

The previous methods assumed available knowledge about pixels belonging to individual objects. Often, we work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be. In such situations, all pixels are candidates for linking and thus have to be accepted or eliminated based on predefined *global* properties. The technique approach in this section is based on whether set of pixels lie on curves of a specified shape. Once detected, these curves form the edges or region boundaries of interest.

Given $\boldsymbol{n}$ points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to find first of all lines determined by every pair of points and then find all subsets of points that are close to particular lines. This approach involves finding $\boldsymbol{\frac{n(n-1)}{2} \sim n^2}$ lines and then performing $\boldsymbol{n\frac{n(n-1)}{2} \sim n^3}$ comparisons of every point to all lines. This is a computationally prohibitive task.

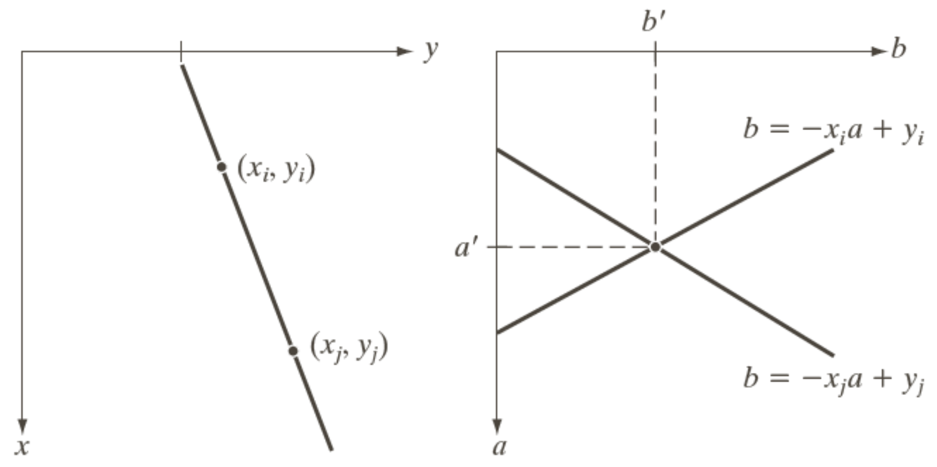
Hough proposed an alternative approach, commonly referred to as *Hough transform*. Consider a point (x_i , y_i) in the xy-plane and the general equation of a line that passes through this point:

$$y_i = a x_i + b$$

Infinitely many lines pass through $\boldsymbol{(x_i , y_i)}$, but they all satisfy the equation $\boldsymbol{y_i = a\, x_i + b}$ for varying values of ***a*** and ***b***. However, writing this equation as:

$$\boldsymbol{b = -x_i\, a + y_i}$$

And considering the ***ab***-plane (also called *parameter space*) yields the equation of a *single* line for a fixed pair $\boldsymbol{(x_i , y_i)}$. Furthermore, a second point $\boldsymbol{(x_j , y_j)}$ also has a line in the parameter space associated with it, and, unless they are parallel, this line intersects the line associated with ***(x_i , y_i)*** at some point ***(a', b')***. In fact, all the points on this line have lines in parameter space that intersect at ***(a', b')***.



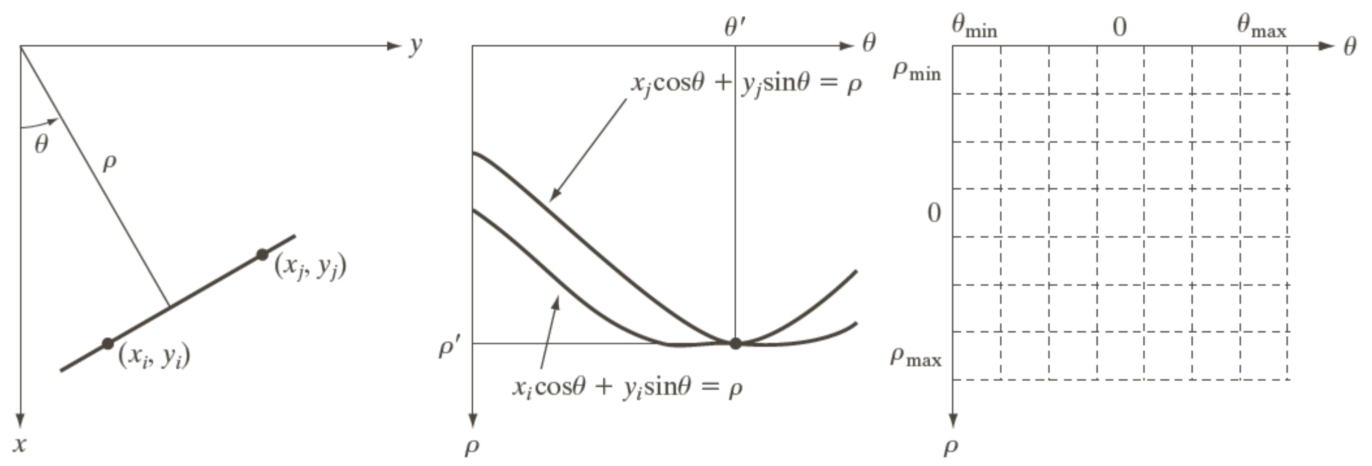
a b

FIGURE 10.31
(a) *xy*-plane.
(b) Parameter space.

In principle, the parameter-space lines corresponding to all points ***(x***$(x_k\ ,\ y_k\ )$ in the ***xy***-plane could be plotted, and the principal lines in that plane could be found by identifying points in parameter space where large numbers of parameter space line intersect. A practical difficulty with this approach, however, is that ***a*** tends to infinity as the lines approaches the vertical direction. To solve this inconvenient, we use the normal representation of a line:

$$x\cos\theta + y\sin\theta = \rho$$

Course 10



a b c

FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.

The computational attractiveness of the Hough transform arises from subdividing the $\boldsymbol{\rho\theta}$ parameter space into so called *accumulator cells*, as Figure 10.32(c) illustrates, where $(\rho_{\min}, \rho_{\max})$ and $(\theta_{\min}, \theta_{\max})$ are the expected ranges of the parameter values: $-90^o \le \theta \le 90^o$ and $-D \le \rho \le D$, where D is the maximum distance between opposite corners in an image. The cell at coordinates $(i\ ,\ j\ )$ with accumulator value $A(i, j)$, corresponds to the square associated with parameter-space coordinates $(\rho_i\ ,\ \theta_j\ )$. In initially, these cells are set to zero.

Then, for every non-background point (x_k, y_k) in the ***xy***-plane, we let θ equal each of the allowed subdivision values on the θ-axis and solve for the corresponding ρ using the equation $\rho = x_k \cos\theta + y_k \sin\theta$. The resulting ρ values are then rounded off to the nearest allowed cell value along the ρ axis. If a choice of θ_p results in solution ρ_q, then we let

$$A(p,q)=A(p,q)+1.$$

At the end of this procedure, a value of P in $A(i,j)$ means that P points in the ***xy***-plane lie on the line $x\cos\theta_j + y\sin\theta_j = \rho_i$.

The number of subdivisions in the $\boldsymbol{\rho\theta}$-plane determines the accuracy of the collinearity of these points. It can be shown that the number of computations in the above described method is linear with respect to $\boldsymbol{n}$, the number of non-background points in the $\boldsymbol{xy}$-plane.

An approach based on the Hough transform for edge-linking is as follows:

1. Obtain a *binary* edge image.
2. Specify subdivisions in the $\boldsymbol{\rho\theta}$-plane.

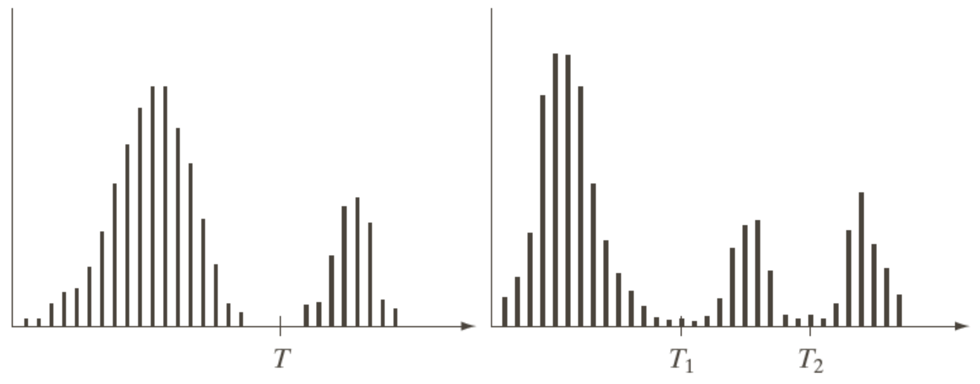
3. Examine the counts of the accumulator cells for high pixel concentration
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

Continuity in this case usually is based on computing the distance between disconnected pixels corresponding to a given accumulator cell. A gap in a line associated with a given cell is bridged if the length of the gap is less than a specified threshold.

Image Segmentation – Thresholding

We discuss techniques for partitioning images directly into regions based on intensity values and/or properties of these values.

Suppose that the intensity histogram in Figure 10.35(a) corresponds to an image, ***f(x,y)***, composed of light objects and a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One way to extract the objects from the background is to select a threshold ***T*** that separates these modes.



a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

Any point ***(x,y)*** in the image for which ***f(x,y)*** > ***T*** is called an *object point;* otherwise, the point is called a *background point*. The segmented image, ***g(x,y)***, is given by:

$$\boldsymbol{g(x,y)} = \begin{cases} \mathbf{1} & \text{if } \boldsymbol{f(x,y) > T} \\ \mathbf{0} & \text{if } \boldsymbol{f(x,y) \le T} \end{cases}$$

where ***T*** is a constant applicable over an entire image, the process given in this equation is referred to as *global thresholding*. When the value of ***T*** changes over an image, we use the term *variable thresholding*. The term *local* or *regional thresholding* is used sometimes to denote variable thresholding in which the value of ***T*** at any point ***(x, y)*** in an image depends on properties of a neighborhood of ***(x, y)*** (for example, the average intensity of the pixels in the neighborhood). If ***T*** depends on the spatial coordinates ***(x, y)*** themselves, then variable thresholding is often referred to as *dynamic* or *adaptive* thresholding.

If in an image we have, for example, two types of light objects on a dark background, *multiple thresholding* is used. The segmented image is given by:

$$\boldsymbol{g(x,y)} = \begin{cases} \boldsymbol{a} & \text{if } \boldsymbol{f(x,y) > T_2} \\ \boldsymbol{b} & \text{if } \boldsymbol{T_1 < f(x,y) \le T_2} \\ \boldsymbol{c} & \text{if } \boldsymbol{f(x,y) \le T_1} \end{cases}$$

where ***a*, *b*,** and ***c*** are any three distinct intensity values. Segmentation problems requiring more than two thresholds are difficult to solve, and better results usually are obtained using other methods.

The success of intensity thresholding is directly related to the width and depth of the valley(s) separating the histogram modes. The key factors affecting the properties of the valley(s) are:

(1) the separation between peaks (the further apart the peaks are, the better the chances of separating the modes)

(2) the noise content in the image (the modes broaden as noise increases)

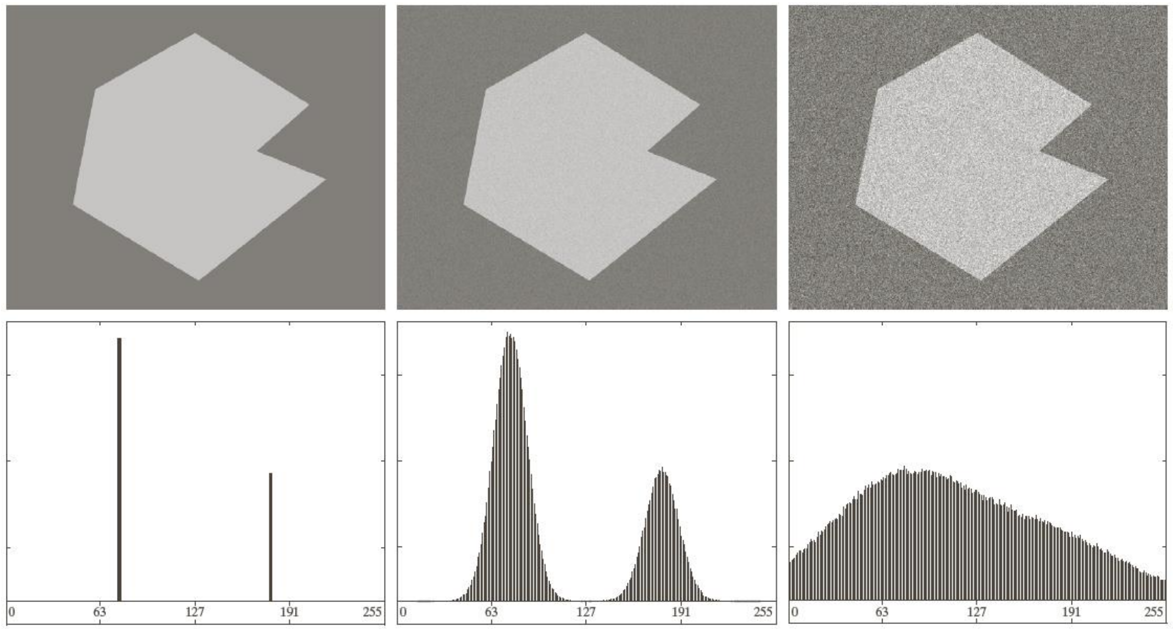
(3) the relative size of objects and background

(4) the uniformity of the illumination source

(5) the uniformity of the reflectance properties of the image

The role of noise in image thresholding

Consider Figure 10.36(a) – the image is free of noise and its histogram has two “spike” modes. Figure 10.36(b) shows the original image corrupted by Gaussian noise of zero mean and a standard deviation of 10 intensity levels. Although the corresponding histogram modes are now broader (Figure 10.36(e)), their separation is large enough so that the depth of the valley between them is sufficient to make the modes easy to separate.



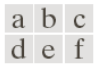


FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Figure 10.36(c) shows the result of corrupting the image with Gaussian noise of zero mean and a standard deviation of 50 intensity levels. As the histogram in Figure 10.36(f) shows, the situation is much more difficult as there is now way to differentiate between two modes.

The role of illumination and reflectance

Figure 10.37 illustrates the effect that illumination can have on the histogram of an image. Figure 10.37(a) is the noisy image form Figure 10.36(b) and Figure 10.37(d) shows its histogram.

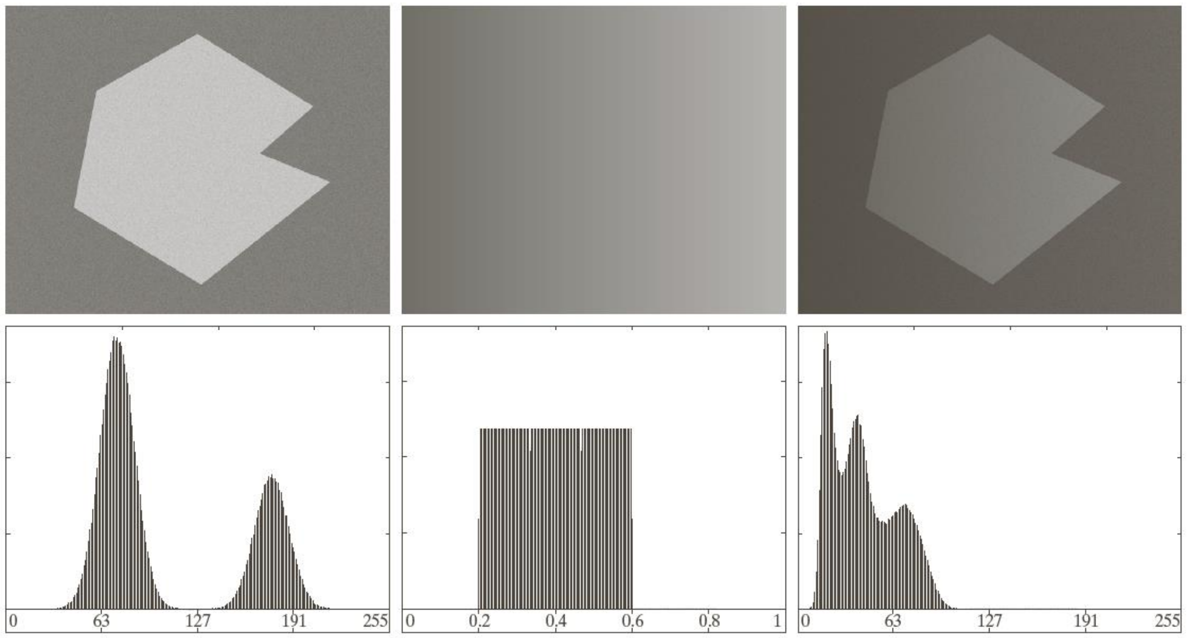


FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

We can illustrate the effects of nonuniform illumination by multiplying the image in Figure 10.37(a) by a variable intensity function, such the intensity ramp in Figure 10.37(b), whose histogram is shown in Figure 10.37(e). Figure 10.37(c) shows the product of the image and this shading pattern. Figure 10.37(f) shows, the deep valley between peaks was corrupted to the point where separation of the modes without additional processing is no longer possible.

Illumination and reflectance play a central role in the success of image segmentation using thresholding or other segmentation techniques. Therefore, controlling these factors when it is possible to do so should be the first step considered in the solution of segmentation problem. There are three basic approaches to the problem when control over these factors is not possible. One is to correct the shading pattern directly. For example, nonuniform (but fixed) illumination can be corrected by multiplying the image by the inverse pattern, which can be obtained by imaging a flat surface of constant intensity. The second approach is to attempt to correct the global shading pattern via processing it. The third approach is to “work around” nonuniformities using variable thresholding.

Basic Global Thresholding

When the intensity distributions of objects and background pixels are sufficiently distinct, it is possible to use a single (*global*) threshold applicable over the entire image. An algorithm capable of estimating automatically the threshold value for each image is required. The following iterative algorithm can be used for this purpose:

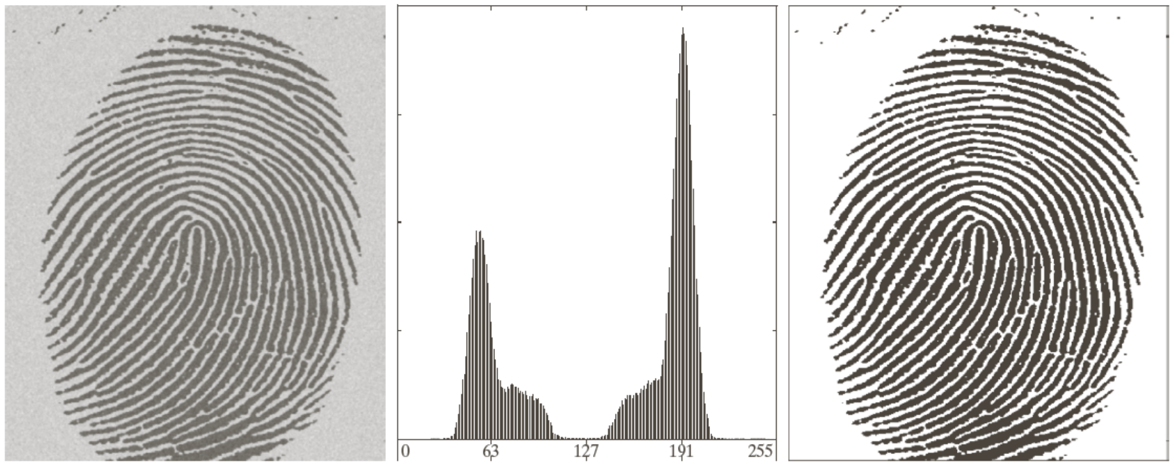
1. Select an initial estimate for the global threshold, $\boldsymbol{T}$.

2. Segment the image using $\boldsymbol{T}$. This will produce two groups of pixels: $\boldsymbol{G_1}$ consisting of all pixels with intensity values $> \boldsymbol{T}$, and $\boldsymbol{G_2}$ consisting of pixels with values $\leq \boldsymbol{T}$.
3. Compute the average (mean) intensity values $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ for the pixels in $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$.
4. Compute a new threshold value:

$$\boldsymbol{T = \frac{1}{2}(m_1 + m_2)}$$

5. Repeat Steps 2 through 4 until the difference between values of $\boldsymbol{T}$ in successive iteration is smaller than a predefined parameter $\boldsymbol{\Delta T}$

This simple algorithm works well in situations where there is a reasonably clear valley between the modes of the histogram related to objects and background. Parameter $\boldsymbol{\Delta T}$ is used to control the number of iterations in situations when speed is an important issue. The initial threshold must be chosen greater than the minimum and less than maximum intensity level in the image. The average intensity of the image is a good initial choice for ***T***.



a b c

FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

Optimum Global Thresholding Using Otsu's Method

Thresholding may be viewed as a statistical-decision theory problem whose objective is to minimize the average error that appears in assigning pixels to two or more groups (also called *classes*). The solution (*Bayes decision rule)* is based on only two parameters: the probability density function (PDF) of the intensity levels of each class and the probability that each class occurs in a given application. Estimating PDFs is not a trivial task, so the problem usually is simplified by making workable assumption about the form of the PDFs, such as assuming that they are Gaussian function.

Otsu's method offers an alternative solution. The method is optimum in the sense that it maximizes the *between-class variance*. The basic idea is that the well-thresholded classes should be distinct with respect to the intensity values of their pixels and, conversely, that a threshold giving the best separation between classes in terms of their intensity values would be the best (optimum) threshold. Otsu's method has the important property that it is based entirely on computations performed of the histogram of an image.

Let ***{0,1, 2, …, L-1}*** denote the ***L*** distinct intensity levels in a digital image of size ***M×N*** pixels, and let $\boldsymbol{n_i}$ denote the number of pixels with intensity ***i***.

$$\boldsymbol{MN}\,(\text{total number of pixels}) = \boldsymbol{n_0 + n_1 + n_2 + \cdots + n_{L-1}}$$

The normalized histogram has components $\boldsymbol{p_i = \frac{n_i}{MN}}$

with:

$$\sum_{i=0}^{L-1} p_i = 1\,, \quad p_i \geq 0.$$

Suppose that we select a threshold $\boldsymbol{T(k)=k}$, $\boldsymbol{0 < k < L\text{-}1}$ and use it to threshold the image into two classes $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ where $\boldsymbol{C_1}$ consists of all pixels in the image with intensity values in the range $\boldsymbol{[0,k]}$ and $\boldsymbol{C_2}$ consists of all pixels in the image with intensity values in the range ***[k+1,L-1]*** . Using this threshold the probability, ***$P_1(k)$*** that a pixel is assigned to class ***C_1*** is given by the cumulative sum:

$$P_1(k) = \sum_{i=0}^{k} p_i$$

This is the probability of class ***C_1*** occurring. Similarly, the probability of class ***C_2*** occurring is:

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i \,.$$

The mean intensity values of the pixels assigned to class C_1 is

$$m_1(k) = \sum_{i=0}^{k} iP(i/C_1) = \sum_{i=0}^{k} iP(C_1/i)\frac{P(i)}{P(C_1)} = \frac{1}{P_1(k)}\sum_{i=0}^{k} ip_i$$

$P(i/C_1)$ is the probability of value i , given that i comes from class C_1. We have used the Bayes' formula:

$$P(A/B) = P(B/A)\frac{P(A)}{P(B)}.$$

$P(C_1/i)=1$ – the probability of C_1 given i (i belongs to C_1).

Similarly, the mean intensity value of the pixels assigned to class ***C*₂** is:

$$m_2(k)=\sum_{i=k+1}^{L-1} i\,P(i/C_2)=\frac{1}{P_2(k)}\sum_{i=k+1}^{L-1} i\,p_i\,.$$

The cumulative mean (average intensity) up to level ***k*** is given by:

$$m(k)=\sum_{i=0}^{k} ip_i$$

and the average intensity of the entire image (the *global* mean) is given by:

$$m_G = \sum_{i=0}^{L-1} i p_i$$

We have:

$$P_1 m_1 + P_2 m_2 = m_G \quad , \quad P_1 + P_2 = 1.$$

In order to evaluate the "goodness" of the threshold at level ***k*** we use the normalized, dimensionless metric:

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

where σ_G^2 is the *global variance*:

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

and σ_B^2 is the *between-class variance*, defined as:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1 P_2 (m_1 - m_2)^2 =$$

$$= \frac{\left(m_G P_1 - m\right)^2}{P_1(1 - P_1)}$$

From the above formula, we see that the farther the two means $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ are from each other the larger $\boldsymbol{\sigma_B^2}$ will be, indicating that the between-class variance is a measure of *separability* between classes. Because $\boldsymbol{\sigma_G^2}$ is a constant, it follows that $\boldsymbol{\eta}$ *also is a measure* of separability, and maximizing this metric is equivalent to maximizing $\boldsymbol{\sigma_B^2}$.

The objective then is to determine the threshold value $\boldsymbol{k}$ that maximizes the between-class variance.

We have:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

$$\sigma_B^2(k) = \frac{\left[m_G P_1(k) - m(k)\right]}{P_1(k)\left[1 - P_1(k)\right]}$$

The optimum threshold is the value k^* that maximizes $\sigma_B^2(k)$

$$\sigma_B^2(k^*) = \max\{\ \sigma_B^2(k)\ ;\ \ 0 \le k \le L-1\ ,\ \ k - \text{integer}\}.$$

If the maximum exists for more than one value of k, it is customary to average the various values of k for which $\sigma_B^2(k)$ is maximum.

Once $\boldsymbol{k}^*$ has been obtained, the input image is segmented as:

$$\boldsymbol{g(x,y)=\begin{cases}1 & \text{if } f(x,y)>k^* \\ 0 & \text{if } f(x,y)\le k^*\end{cases}}$$

The metric $\boldsymbol{\eta(k^*)}$ can be used to obtain quantitative estimate of the separability of classes.

$$\boldsymbol{0\le\eta(k^*)\le 1}$$

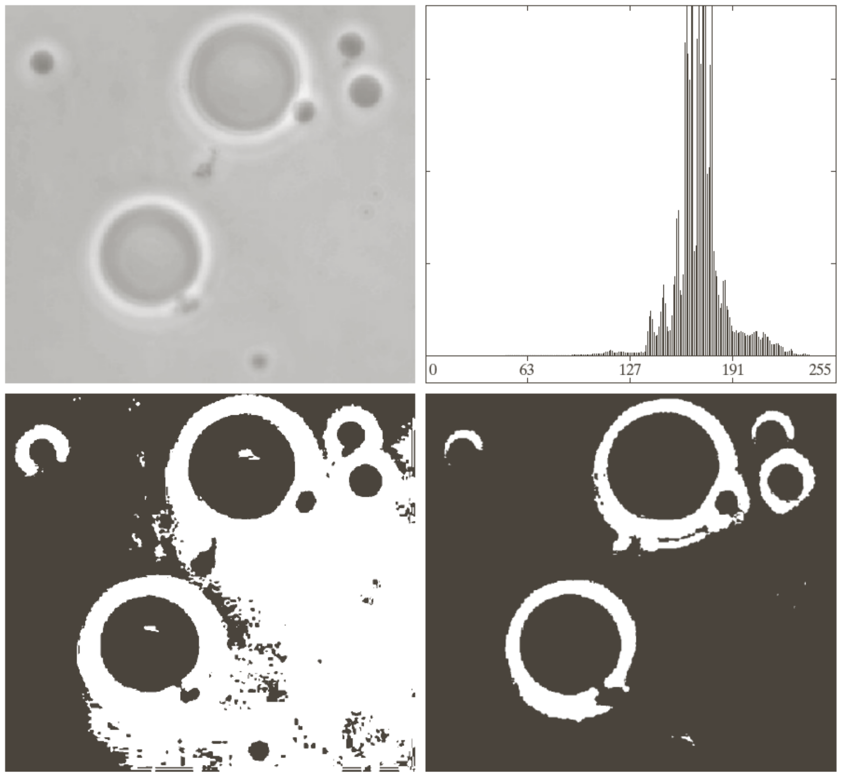
The lower bound is attainable only by images with a single, constant intensity level, the upper bound is attainable only by 2-valued images with intensities equal to $\boldsymbol{0}$ and $\boldsymbol{L-1}$.

Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image, $\boldsymbol{p_i\ ,\ i=0,1,2,\ldots,L-1}$
2. Compute the cumulative sums, $\boldsymbol{P_1(k),k=0,1,2,\ldots,L-1}$
3. Compute the cumulative means, $\boldsymbol{m(k),k=0,1,\ldots,L-1}$
4. Compute the global intensity mean, $\boldsymbol{m_G}$

5. Compute the between-class variance, $\sigma_B^2(k)$, $k=0,1,\ldots,L-1$
6. Obtain the Otsu threshold, k^*, as the value of k for which $\sigma_B^2(k)$ is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected
7. Obtain the separability measure, $\eta(k^*)$

Course 10

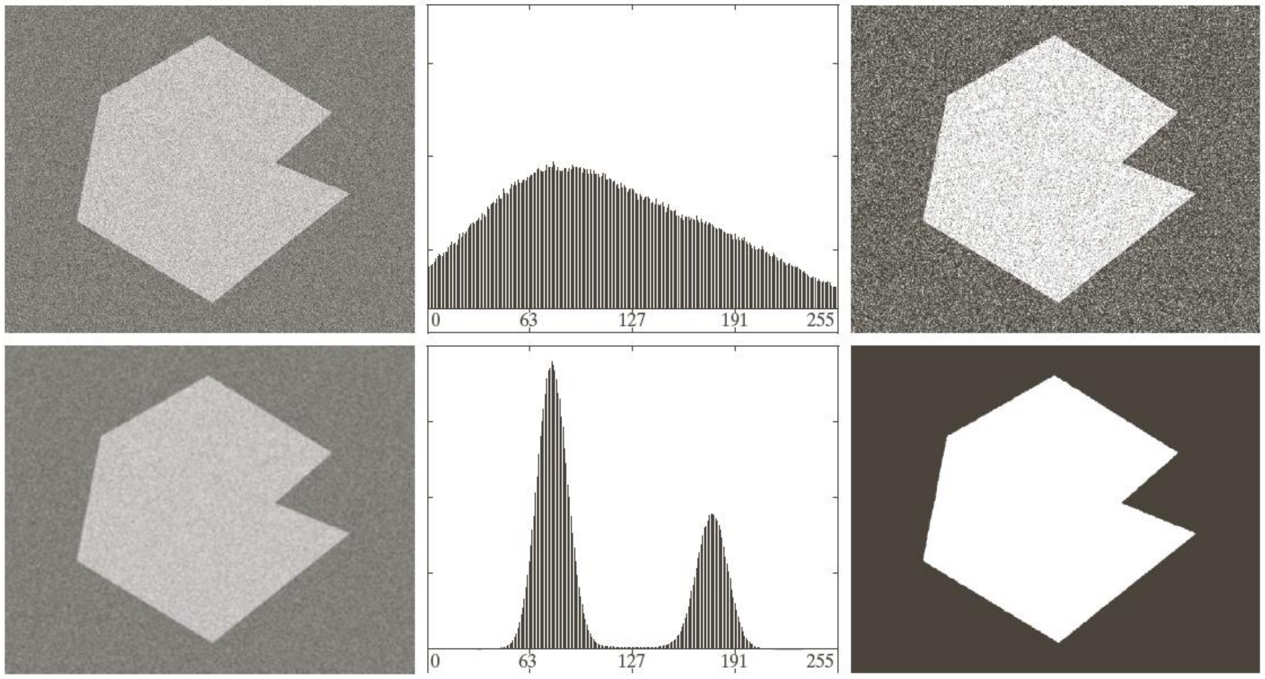


a b
c d

FIGURE 10.39 (a) Original image. (b) Histogram (high peaks were clipped to highlight details in the lower values). (c) Segmentation result using the basic global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

Noise can turn a simple thresholding problem into an unsolvable one. When noise cannot be reduced at the source, and thresholding is the segmentation method used, a technique that often enhances performances is to smooth the image before thresholding it.

Course 10



a b c
d e f

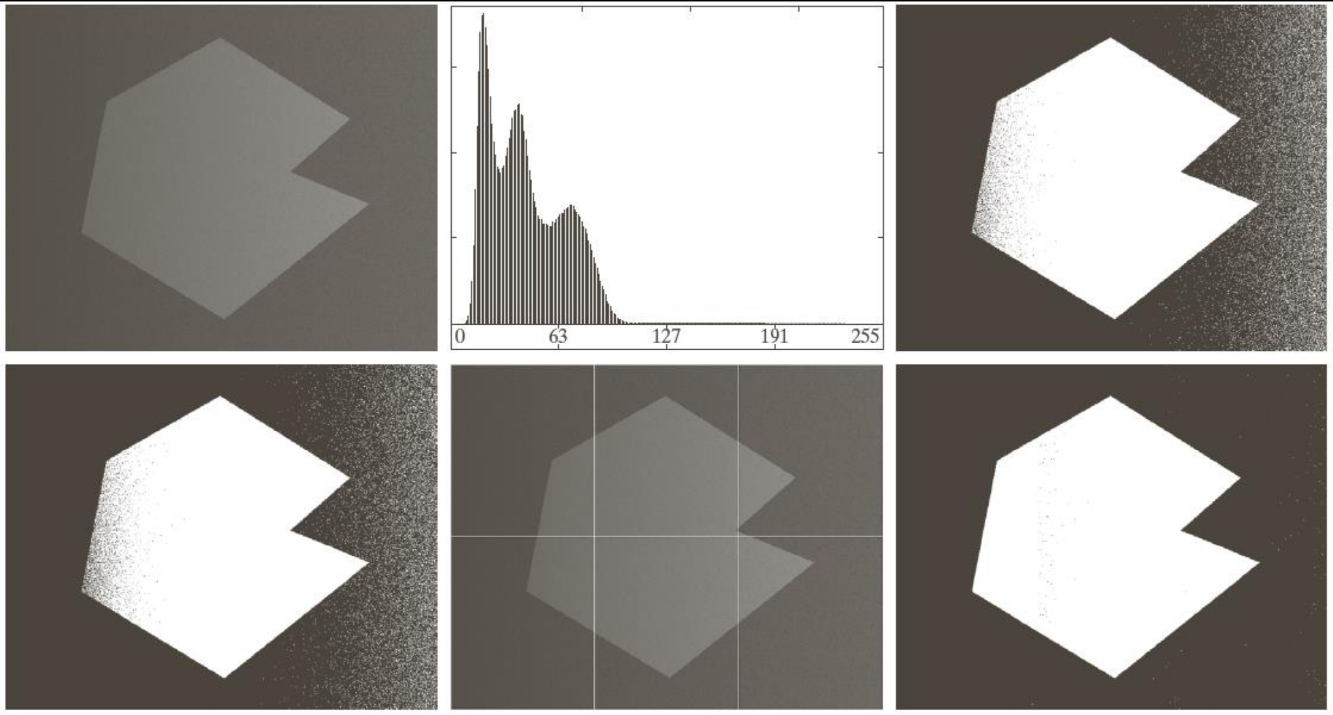
FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5 × 5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

Variable Thresholding

Image partitioning

One of the simplest approaches to variable thresholding is to subdivide an image into nonoverlapping rectangles. This approach is used to compensate for non-uniformities in illumination and/or reflectance. The rectangles are chosen small enough so that the illumination of each is approximately uniform.

Course 10



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

Image subdivision generally works well when the objects of interest and the background occupy regions of reasonably comparable size. When this is not the case, the method fails because of the likelihood of subdivisions containing only object or background pixels.

Variable thresholding based on local image properties

A more general approach than the image subdivision method is to compute a threshold at every point (x,y) in the image based on one or more specified properties computed in a neighborhood of (x, y).

We illustrate the basic approach to local thresholding by using the standard deviation and mean of the pixels in a neighborhood of every point in an image. Let σ_{xy} and m_{xy} denote the standard deviation and mean value of a set of pixels contained in a neighborhood $\boldsymbol{S_{xy}}$ centered at coordinates ***(x, y)*** in an image.

The following are common forms of variable, local thresholds

$$T_{xy} = a\sigma_{xy} + bm_{xy} \ , \quad a, b \geq 0$$

$$T_{xy} = a\sigma_{xy} + bm_G \ , \quad m_G \text{ - global image mean}.$$

The segmented image is computed as:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{if } f(x,y) \leq T_{xy} \end{cases}.$$

Significant improvement can be obtained in local thresholding by using predicates based on the parameters computed in the neighborhood of ***(x, y)***:

$$\boldsymbol{g(x,y)} = \begin{cases} \mathbf{1} & \text{if } \boldsymbol{Q}(\text{local parameters}) \text{ is true} \\ \mathbf{0} & \text{if } \boldsymbol{Q}(\text{local parameters}) \text{ is false} \end{cases}$$

Where ***Q*** is a *predicate* based on parameters computed using the pixels in neighborhood ***S**xy*.

$$\boldsymbol{Q(\sigma_{xy}, m_{xy})} = \begin{cases} \text{true} & \text{if } \boldsymbol{f(x,y) > a\sigma_{xy}} \text{ AND } \boldsymbol{f(x,y) > bm_{xy}} \\ \text{false} & \text{otherwise} \end{cases}$$

Multivariable Thresholding

In some cases, a sensor can make available more than one variable to characterize each pixel in an image, and thus allow *multivariable thresholding*. A notable example is color imaging where red (R), green (G), and blue (B) components are used to form a composite color image. In this case, each "pixel" is characterized by three values, and can be represented as a 3-D vector $\boldsymbol{z} = (z_1, z_2, z_3)^T$ whose components are the RGB colors at a point.

These 3D points often are referred to as *voxels*, to denote *volumetric* elements, as opposed to *image* elements.

Multivariable thresholding may be viewed as a distance computation. Suppose that we want to extract from a color image all regions having a specified color range: say, reddish hues. Let ***a*** denote the average reddish color in which we are interested.

One way to segment a color image based on this parameter is to compute a distance measure, ***D(z,a)*** between an arbitrary color point ***z*** and the average color ***a***. Then we segment the input image:

$$g = \begin{cases} 1 & \text{if } D(z,a) < T \\ 0 & \text{otherwise} \end{cases} \quad , \quad T \text{ is a threshold}$$

D(z,a) – *Euclidian distance*

$$D(z,a) = \left[(z-a)^T (z-a) \right]^{\frac{1}{2}}$$

Mahalanobis distance

$$\boldsymbol{D(z,a)=\left[(z-a)^{T}C^{-1}(z-a)\right]^{\frac{1}{2}}}$$

where $\boldsymbol{C}$ is the covariance matrix of the $\boldsymbol{z}$s.

Region-Based Segmentation

Region growing

Region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria for growth. The basic approach is to start with a set of “seed” points and form these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as specific ranges of intensity colors).

Selecting a set of one or more starting points often can be based on the nature of the problem. When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available.

Another problem in region growing is the formulation of a stopping rule. Region growth should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as intensity values, texture, and color are local in nature and do not take into account the "history" of region growth.

Additional criteria that increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far, and the shape of the region being grown.

Let ***f(x,y)*** denote an input image array, ***S(x,y)*** denote a *seed* array containing ***1***s at the locations of seed points and ***0***s elsewhere, ***Q*** denote a predicate to be applied at each pixel location ***(x, y)***.

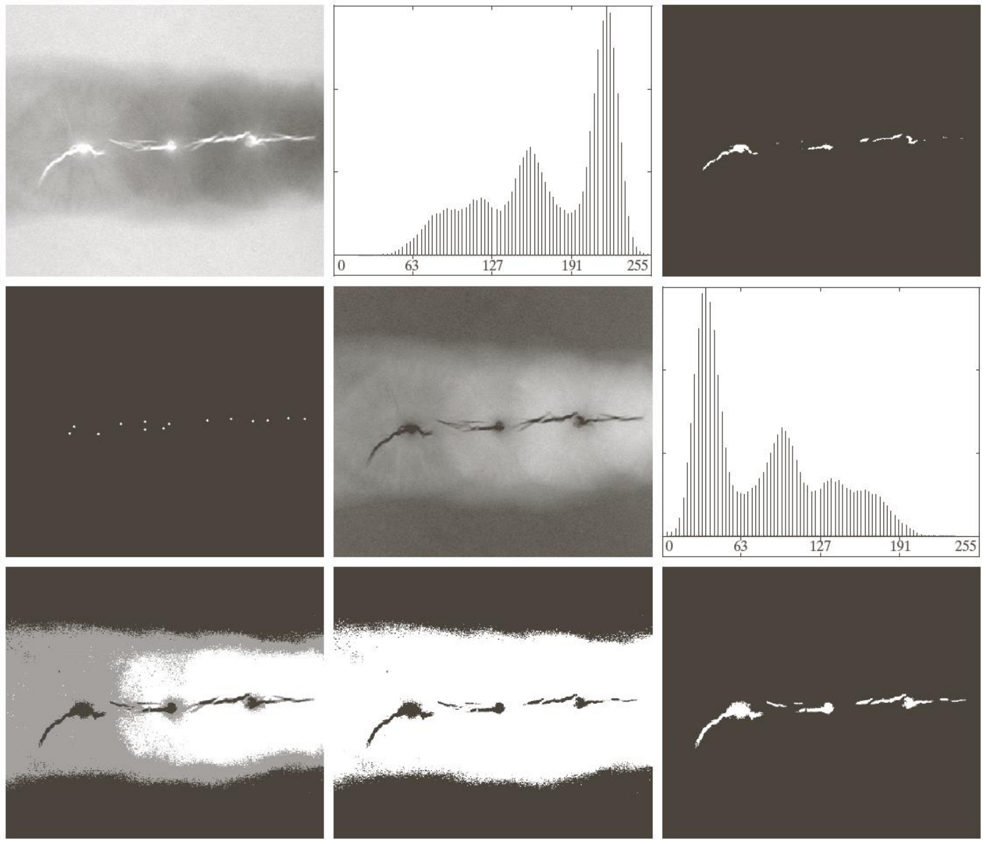
Arrays ***f*** and ***S*** are assumed to be of the same size. A basic region-growing algorithm based on ***8***-connectivity may be stated as follows:

1. Find all connected components in ***S(x,y)*** and erode each connected component to one pixel; label all such pixels found as ***1***. All other pixels in ***S*** are labeled ***0***.
2. Form an image f_Q such that, at a pair of coordinates ***(x,y)***, let $f_Q(x,y)=1$ if the input image satisfies the given predicate, ***Q,*** at those coordinates, otherwise, let $f_Q(x,y)=0$.

3. Let ***g*** be an image formed by appending to each seed point in ***S*** all the ***1***-valued points in f_Q that are ***8***-connected to that seed point.
4. Label each connected component in ***g*** with a different region label. This is the segmented region obtained by region growing.

$$\boldsymbol{Q} = \begin{cases} \text{TRUE} & \text{if the absoulte difference of the intensities} \\ & \text{between the seed and the pixel at } (\boldsymbol{x}, \boldsymbol{y}) \text{ is } \leq \boldsymbol{T} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Course 10



a b c
d e f
g h i

FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

Region Splitting and Merging

The method used in this case is to subdivide an image initially into a set of arbitrary, disjoint regions and then merge and/or split the regions in an attempt to satisfy the condition of segmentation.

Let ***R*** represent the entire image region and select a predicate ***Q***. One approach for segmenting ***R*** is to subdivide it successively into smaller and smaller quadrant regions so that, for any region $\boldsymbol{R_i}$, $\boldsymbol{Q(R_i)=TRUE}$. We start with the entire region. If ***Q(R)=FALSE***, we divide the image into quadrants. If ***Q*** is False for any quadrant, we subdivide the quadrant into subquadrant, and so on. This particular splitting technique has a convenient representation in the form of so-called *quadtrees*, that are trees in which each node has exactly four descendants. The images corresponding to the nodes of a quadtree sometimes are called *quadregions* or *quadimages*. Note that the root of the tree corresponds to the entire image and that each node corresponds to the subdivision of a node into four descendant nodes.

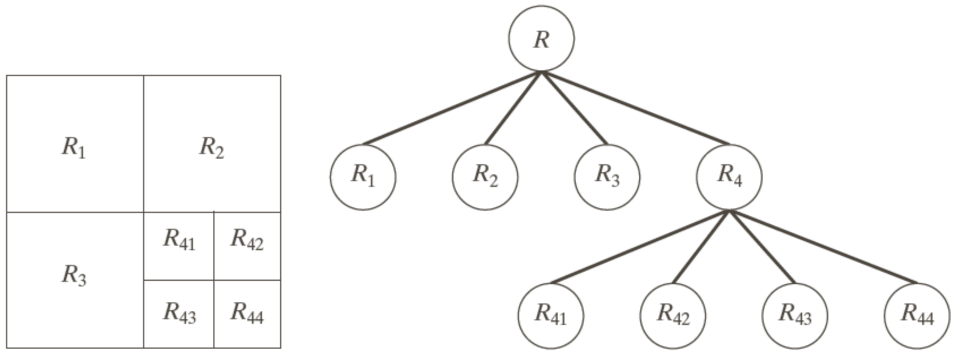


FIGURE 10.52 (a) Partitioned image. (b) Corresponding quadtree. R represents the entire image region.

If only splitting is used, the final partition normally contains adjacent region with identical properties. Satisfying the constraints of segmentation requires merging only adjacent regions whose combined pixels satisfy the predicate ***Q***. That is, two adjacent regions $\boldsymbol{R_j}$ and $\boldsymbol{R_k}$ are merged only if $\boldsymbol{Q(R_j \cup R_k) = TRUE}$.

The procedure described above can be summarized as follows

1. Split into four quadrants any region $\boldsymbol{R_i}$ for which $\boldsymbol{Q(R_i) = TRUE}$
2. When no further splitting is possible, merge any adjacent regions $\boldsymbol{R_j}$ and $\boldsymbol{R_k}$ for which $\boldsymbol{Q(R_j \cup R_k) = TRUE}$
3. Stop when no further merging is possible.

It is customary to specify a minimum quadregions size beyond which no further splitting is carried out.

Numerous variations of the preceding basic theme are possible. For example, a significant simplification results if in Step 2 we allow merging for any two adjacent regions if each one satisfies the predicate individually. This results in a much simpler (and faster) algorithm, because testing the predicate is limited to individual quadregions.

Course 10

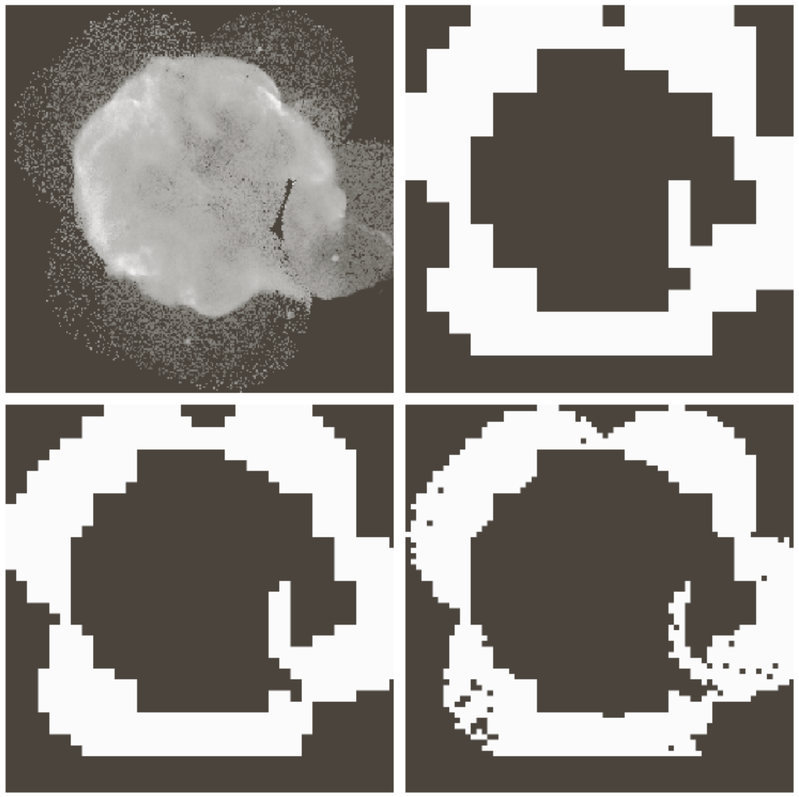


FIGURE 10.53 (a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b)–(d) Results of limiting the smallest allowed quadregion to sizes of 32×32, 16×16, and 8×8 pixels, respectively. (Original image courtesy of NASA.)

$$Q = \begin{cases} \boldsymbol{TRUE} & \text{if } \boldsymbol{\sigma > a \; AND \;\; 0 < m < b} \\ \boldsymbol{FALSE} & \text{otherwise} \end{cases}$$

Where $\boldsymbol{m}$ and $\boldsymbol{\sigma}$ are the mean and the standard deviation of the pixels in a quadregion, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are constants.

Image Compression

Image compression is the art and science of reducing the amount of data required to represent an image.

Consider a two-hour standard definition (SD) television movie using $\boldsymbol{720\times480\times24}$ bit pixel arrays. A digital movie is a sequence of video frames in which each frame is a full-color still image. Because video players must display the frames sequentially at rates 30 fps (frames per second), SD digital video must be accessed at:

$$\mathbf{30}\frac{\text{frames}}{\text{sec}}\times\mathbf{(720\times 480)}\frac{\text{pixels}}{\text{frame}}\times\mathbf{3}\frac{\text{bytes}}{\text{pixel}}=\mathbf{31.104.000}\,\text{bytes/sec}$$

and a two-hour movie consist of

$$\mathbf{31.104.000}\frac{\text{bytes}}{\text{sec}}\times\mathbf{(60^2)}\frac{\text{sec}}{\text{hour}}\times 2\text{hours}\ \simeq \mathbf{2.24\times 10^{11}}\text{bytes}(\mathbf{224}\text{GB})$$

To put a two-hour movie on a DVD, each frame must be compressed by a factor of ***26.3(***on average). The compression must be even higher for *high definition* (HD) *television* where image resolution reach $\boldsymbol{1920\times 1080\times 24}$ bit/image.

Fundamentals

The term *data compression* refers to the process of reducing the amount of data required to represent a given quantity of information. *Data* and *information* are not the same thing; data are the means by which information is expressed. Because various amounts of data can be used to represent the same amount of information, representations that contain irrelevant or repeated information are said to contain *redundant data*.

Let ***b*** and ***b'*** denote the number of bits in two representations of the same information, the *relative data redundancy* ***R*** of the representation with ***b*** bits is:

$$R = 1 - \frac{1}{C}$$

Where ***C***, commonly called the *compression ratio* is

$$C = \frac{b}{b'}$$

If ***C=10*** - the larger representation has ***10*** bits of data for every ***1*** bit of data in the smaller representation. The corresponding relative data redundancy of the larger representation is ***0.9*** (***R=0.9***), indicating that ***90%*** of its data is redundant.

In the context of digital image compression, ***b*** usually is the number of bits needed to represent an image as a 2-D array of intensity values. Two-dimensional intensity arrays (far from optimal) suffer from three principal types of data redundancies:

1. *Coding redundancy*. A *code* is a system of symbols (letters, numbers, bits…) used to represent a body of information or set of events. Each piece of information or event is assigned a sequence of *code symbols*, called a *code word*. The number of symbols in each code word is its *length*. The *8-bit* codes that are used to represent the intensities in most 2-D intensity arrays contain more bits than are needed to represent the intensities.

2. *Spatial* and *temporal redundancy*. Because the pixels of most 2-D intensity arrays are correlated spatially (i.e. each pixel is similar to or dependent on neighboring pixels), information is unnecessarily replicated in the representations of correlated pixels. In a video sequence, temporally correlated pixels (i.e., those similar to or dependent on pixels in nearby frames) also duplicate information.

3. *Irrelevant information.* Most 2-D intensity arrays contain information that is ignored by the human eye. This information is redundant in the sense that it is not used.

Compression is achieved when one or more redundancy is reduced or eliminated.

a b c

FIGURE 8.1 Computer generated 256 × 256 × 8 bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)