Thresholding

Introduction

Segmentation involves separating an image into regions (or their contours) corresponding to objects. We usually try to segment regions by identifying common properties. Or, similarly, we identify contours by identifying *differences* between regions (edges).

The simplest property that pixels in a region can share is intensity. So, a natural way to segment such regions is through *thresholding*, the separation of light and dark regions.

Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one. (What you want to do with pixels at the threshold doesn't matter, as long as you're consistent.) If $g(x, y)$ is a thresholded version of $f(x, y)$ at some global threshold T.
g is equal to 1 if $f(x, y) \geq T$ and zero otherwise.

Problems with Thresholding

The major problem with thresholding is that we consider only the intensity, not any relationships between the pixels. There is no guarantee that the pixels identified by the thresholding process are contiguous.

We can easily include extraneous pixels that aren't part of the desired region, and we can just as easily miss isolated pixels within the region (especially near the boundaries of the region). These effects get worse as the noise gets worse, simply because it's more likely that a pixels intensity doesn't represent the normal intensity in the region.

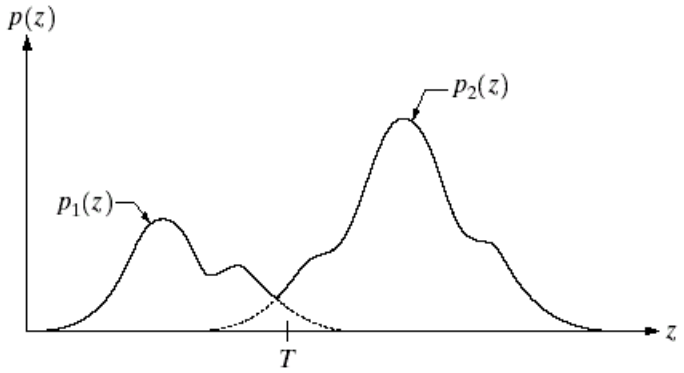
When we use thresholding, we typically have to play with it, sometimes losing too much of the region and sometimes getting too many extraneous background pixels. (Shadows of objects in the image are also a real pain—not just where they fall across another object but where they mistakenly get included as part of a dark object on a light background.)

Optimal Global Thresholding

- A threshold is said to be globally optimal if the number of misclassified pixels is minimum
  - Histogram is bimodal (object and background)
  - Ground truth is known OR the histograms of the object and the background are known

Optimal Global Thresholding

FIGURE 10.32 Gray-level probability density functions of two regions in an image.



- If the individual (normalized) histograms are known as $p_1(z)$ and $p_2(z)$
  - The (normalized) histogram of the overall image is $P_1p_1(z)+P_2p_2(z)$ where $P_1=N_1/(N_1+N_{2)}$ and $P_2=N_2/(N_1+N_2)$ Notice that $P_1+P_2=1$

Probability of Error

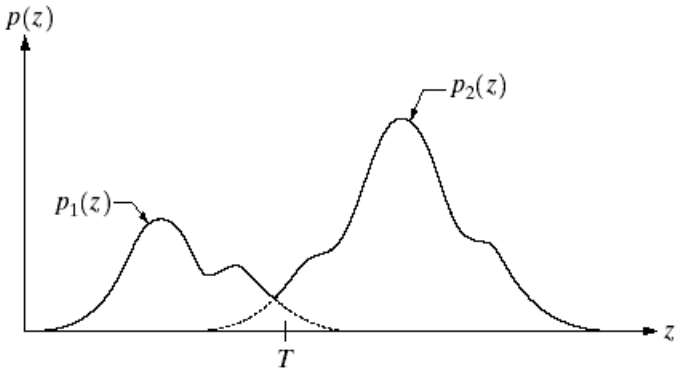


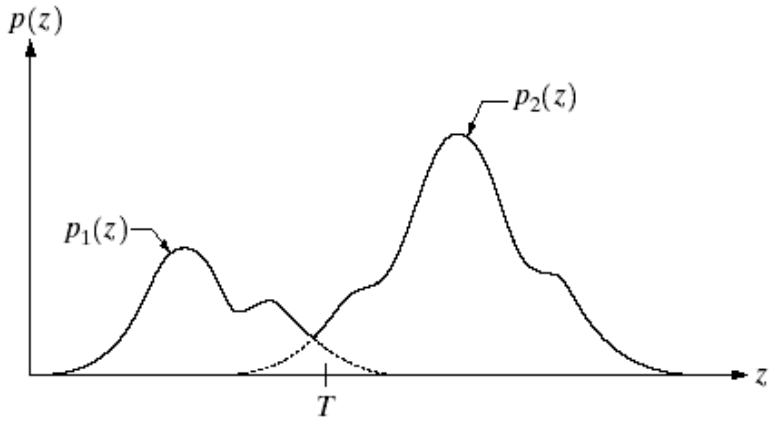
FIGURE 10.32 Gray-level probability density functions of two regions in an image.

- Probability of erroneously classifying a class 1 pixel to class 2 is
  - $$E_2(T) = \int_T^{\infty} p_1(z)dz$$
- Probability of erroneously classifying a class 2 pixel to class 1 is
  - $$E_1(T) = \int_{-\infty}^{T} p_2(z)dz$$
- Overall probability of error is then
  - $E(T)=P_2E_1(T)+P_1E_2(T)$

Minimization of Probability of Error

- Let E'(T)=0 we have $P_1p_1(T)=P_2p_2(T)$
  - If $p_1(T)$, $p_2(T)$ are known, T can be determined
  - If $P_1=P_2$ (# of object pixels = # of background pixels), the optimum T is where the curves intersect
  - What if P_1 is larger? P_2 is larger?

FIGURE 10.32 Gray-level probability density functions of two regions in an image.



A Special Case

- An analytical expression for T is available if both normalized histograms can be modeled as Gaussian distributions
  - $$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}}$$
  - T must satisfy the quadratic equation

$$AT^2 + BT + C = 0$$

where

$$\mathrm{A} = \sigma_1^2 - \sigma_2^2, \mathrm{B} = 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)$$

$$C = \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2 2\sigma_2^2 \ln(\sigma_2 P_1 / \sigma_1 P_2)$$

Parameter Estimation

- Parameters involved in an assumed functional form can be estimated using a minimization of mean-square-error approach

$$e_{ms} = \frac{1}{n}\sum_{i=1}^{n}[\,p(z_i) - h(z_i)\,]^2$$

- The goal is to search for the parameters that minimizes the above quantity

Minimum Error Thresholding

Assume object pixels are distributed (histogram) according to $p_O(x)$ and the background pixels by $p_B(x)$. The error is misclassifying object pixels is

$$\int_{-\infty}^{t} p_O(x)dx$$

And misclassifying background pixels as object pixels is

$$\int_{t}^{\infty} p_B(x)dx$$

Let θ be the fraction of pixels in the object. Then the total error is

$$E(t) = \theta \int_{-\infty}^{t} p_O(x)dx + (1-\theta)\int_{t}^{\infty} p_B(x)dx$$

To find the minimum we differentiate

$$0 = \frac{\partial E}{\partial t} = \theta\, p_O(t) - (1-\theta)\, p_B(t)$$

or

$$\theta\, p_O(t) = (1-\theta)\, p_B(t)$$

Example:

$$p(x) = \frac{3}{4a^3}\left[a^2 - (x-b)^2\right] \quad \text{for } b-a \le x \le b+a$$

$$0 \quad \text{otherwise}$$

$$\theta = \frac{8}{9}$$

a=1 b=5 for background

a=2 b=7 for object

$$\mathrm{p_B}(t) = \frac{3}{4}\left(-t^2 - 24 + 10t\right)$$

$$\mathrm{p_O}(t) = \frac{3}{32}\left(-t^2 - 45 + 14t\right)$$

$$\frac{1}{9}\frac{3}{4}\left(-t^2 - 24 + 10t\right) = \frac{8}{9}\frac{3}{32}\left(-t^2 - 45 + 14t\right)$$

so

$$t = \frac{21}{4} = 5.25$$

Local Thresholding

Another problem with global thresholding is that changes in illumination across the scene may cause some parts to be brighter (in the light) and some parts darker (in shadow) in ways that have nothing to do with the objects in the image.

We can deal, at least in part, with such uneven illumination by determining thresholds locally. That is, instead of having a single global threshold, we allow the threshold itself to smoothly vary across the image.

Another way of dealing with illumination is to consider

$$g(x, y) = r(x, y) * i(x, y)$$
$$\ln g(x, y) = \ln r(x, y) + \ln i(x, y)$$

Hysteresis Thresholding

Take into account neighbors.

1. Choose two thresholds T_{high} and T_{low}
2. If $T > T_{\text{high}}$ the pixel is in the body. If $T < T_{\text{low}}$ the pixel is in the background.
3. If $T_{\text{low}} < T < T_{\text{high}}$ the pixel is in the body only if a neighbor is already in the body
4. Iterate

Automated Methods for Finding Thresholds

To set a global threshold or to adapt a local threshold to an area, we usually look at the histogram to see if we can find two or more distinct modes—one for the foreground and one for the background.
Recall that a histogram is a probability distribution:

$$p(g)=ng\ /n$$

That is, the number of pixels *ng* having grayscale intensity *g* as a fraction of the total number of pixels *n*. Here are five different ways to look at the problem:

Known Distribution

If you know that the object you're looking for is brighter than the background and occupies a certain fraction 1/*p* of the image, you can set the threshold by simply finding the intensity level such that the desired percentage of the image pixels are below this value. This is easily extracted from the cumulative histogram:

$$c(g)= p(g)$$

Simply set the threshold *T* such that $c(T)=1/p$. If you're looking for a dark object on a light background, $c(T)=1-\frac{1}{p}$

Finding Peaks and Valleys

One extremely simple way to find a suitable threshold is to find each of the modes (local maxima) and then find the valley (minimum) between them.

While this method appears simple, there are two main problems with it:

1. The histogram may be noisy, thus causing many local minima and maxima. To avoid this, the histogram is smoothed before trying to find separate modes.
2. The sum of two separate distributions, each with their own mode, may not produce a distribution with two distinct modes.

Clustering (K-Means Variation)

Another way to look at the problem is that we have two groups of pixels, one with one range of values and one with another. What makes thresholding difficult is that these ranges usually overlap. What we want to do is to minimize the error of classifying a background pixel as a foreground one or vice versa. To do this, we try to minimize the area under the histogram for one region that lies on the other region's side of the threshold. The problem is that we don't have the histograms for each region, only the histogram for the combined regions. (If we had the regions, why would we need to do segmentation?)

Understand that the place of minimum overlap (the place where the misclassified areas of the distributions are equal) is *not* is not necessarily where the valley occurs in the combined histogram. This occurs, for example, when one cluster has a wide distribution and the other a narrow one.

One way that we can try to do this is to consider the values in the two regions as two *clusters*. **The idea is to pick a threshold such that each pixel on each side of the threshold is closer in intensity to the mean of all pixels on that side of the threshold than the mean of all pixels on the other side of the threshold**.
In other words, let $\mu_B(T)$ be the mean of all pixels less than the threshold and $\mu_O(T)$ be the mean of all pixels greater than the threshold. We want to find a threshold such that the following holds:

$$\forall g \geq T : | g - \mu_B(T) | > | g - \mu_O(T) |$$

and

$$\forall g < T : | g - \mu_B(T) | < | g - \mu_O(T) |$$

The basic idea is to start by estimating $\mu_B(T)$ as the average of the four corner pixels (assumed to be background) and $\mu_O(T)$ as the average of everything else. Set the threshold to be halfway between $\mu_B(T)$ and $\mu_O(T)$ (thus separating the pixels according to how close their intensities are to $\mu_B(T)$ and $\mu_O(T)$ respectively). Now, update the estimates of $\mu_B(T)$ and $\mu_O(T)$ respectively by actually calculating the means of the pixels on each side of the threshold. This process repeats until the algorithm converges.
This method works well if the spreads of the distributions are approximately equal, but it does not handle well the case where the distributions have differing variance.

Iterative threshold selection

1. Initial guess of body – e.g. background is four corners
2. Compute mean background and mean object at step t

$$\mu^t{}_B = \frac{\sum_{i,j \in background} g(i,j)}{\#\text{background pixels}} \qquad \mu^t{}_O = \frac{\sum_{i,j \in objects} g(i,j)}{\#object \text{ pixels}}$$

3. Set

$$T^{t+1} = \frac{\mu^t{}_B + \mu^t{}_O}{2}$$

4. If $T^{t+1} = T^t$ halt otherwise go to #2

Clustering (The Otsu Method)

Another way of accomplishing similar results is to set the threshold so as to try to make each cluster as tight as possible, thus (hopefully!) minimizing their overlap. Obviously, we can't change the distributions, but we can adjust where we separate them (the threshold). As we adjust the threshold one way, we increase the spread of one and decrease the spread of the other. The goal then is to select the threshold that minimizes the combined spread.

We can define the *within-class* variance as the weighted sum of the variances of each cluster:

$$\sigma^2_{\text{within}}(T) = n_B(T)\sigma^2{}_B(T) + n_O(T)\sigma^2{}_O(T)$$

where

$$n_B(T) = \sum_{i=0}^{T-1} p(i)$$

$$n_O(T) = \sum_{i=T}^{N-1} p(i)$$

$\sigma^2{}_B(T) =$ the variance of the pixels in the background (below threshold)

$\sigma^2{}_O(T) =$ the variance of the pixels in the foreground (above threshold)

and [0,N -1]is the range of intensity levels.

Computing this within-class variance for each of the two classes for each possible threshold involves a lot of computation, but there's an easier way.

If you subtract the within-class variance from the total variance of the combined distribution, you get something called the between-class variance:

$$\sigma^2_{\text{between}}(T) = \sigma^2 - \sigma^2{}_{Within} = n_B(T)\left[\mu_B(T) - \mu\right]^2 + n_O(T)\left[\mu_O(T) - \mu\right]^2$$

where σ^2 is the combined variance and μ is the combined mean. Notice that the between-class variance is simply the weighted variance of the cluster means themselves around the overall mean. Substituting $\mu = nB\,(T)\mu B\,(T) + nO\,(T)\mu O\,(T)$ and simplifying, we get

$$\sigma^2_{\text{between}}(T) = n_B(T)n_O(T)\left[\mu_B(T) - \mu_O(T)\right]^2$$

So, for each potential threshold T we

1. Separate the pixels into two clusters according to the threshold.
2. Find the mean of each cluster.
3. Square the difference between the means.
4. Multiply by the number of pixels in one cluster times the number in the other.

This depends only on the difference between the means of the two clusters, thus avoiding having to calculate differences between individual intensities and the cluster means. The optimal threshold is the one that maximizes the between-class variance (or, conversely, minimizes the within-class variance).

This still sounds like a lot of work, since we have to do this for each possible threshold, but it turns out that the computations aren't independent as we change from one threshold to another. We can update $n_B(T)$, $n_O(T)$, and the respective cluster means $\mu_B(T)$ and $\mu_O(T)$ as pixels move from one cluster to the other as T increases. Using simple recurrence relations we can update the between-class variance as we successively test each threshold:

$$n_B(T+1)=n_B(T)+n_T$$
$$n_O(T+1)=n_O(T)-n_T$$
$$\mu_B(T+1)=\frac{\mu_B(T)n_B(T)+n_T T}{n_B(T+1)}$$
$$\mu_O(T+1)=\frac{\mu_O(T)n_O(T)-n_T T}{n_O(T+1)}$$

This method is called the *Otsu* method

Mixture Modeling

Another way to minimize the classification error in the threshold is to suppose that each group is Gaussian-distributed. Each of the distributions has a mean (μ_B and μ_O respectively) and a standard deviation (σ_B and σ_O respectively) *independent of the threshold we choose*:

$$h_{\text{model}}(g) = n_B e^{-(g-\mu_B)^2/2\sigma^2_B} + n_O e^{-(g-\mu_O)^2/2\sigma^2_O}$$

Whereas the Otsu method separated the two clusters according to the threshold and tried to optimize some statistical measure, *mixture modeling* assumes that there already exists two distributions and we must find them. Once we know the parameters of the distributions, it's easy to determine the best threshold.

Unfortunately, we have six unknown parameters ($n_B, n_O, \mu_B, \mu_O, \sigma_B, \sigma_O$), so we need to make some estimates of these quantities.

If the two distributions are reasonably well separated (some overlap but not too much), we can choose an arbitrary threshold T and assume that the mean and

standard deviation of each group approximates the mean and standard deviation of the two underlying populations. We can then measure how well a mix of the two distributions approximates the overall distribution:

$$F = \sum_{0}^{N-1} \left[h_{model}(g) - h_{image}(g) \right]^2$$

Choosing the optimal threshold thus becomes a matter of finding the one that causes the mixture of the two estimated Gaussian distributions to best approximate the actual histogram (minimizes *F*). Unfortunately, the solution space is too large to search exhaustively, so most methods use some form of gradient descent method. Such gradient-descent methods depend heavily on the accuracy of the initial estimate, but the Otsu method or similar clustering methods can usually provide reasonable initial estimates.

Mixture modeling also extends to models with more than two underlying distributions (more than two types of regions). For example, segmenting CT images into gray matter, white matter, and cerebral spinal fluid (CSF).

Multispectral Thresholding

We are interested in a technique for segmenting images with multiple components (color images, Landsat images, or MRI images with T1, T2, and proton-density bands). It works by estimating the optimal threshold in one channel and then segmenting the overall image based on that threshold. We then subdivide each of these regions independently using properties of the second channel. We repeat it again for the third channel, and so on, running through all channels repeatedly until each region in the image exhibits a distribution indicative of a coherent region (a single mode).
e.g.
1. Compute histogram for each channel separately.
2. Find the peak in each histogram
   Select **two** thresholds corresponding to some valley on each side of these peaks.
   Segment image into two regions. One between these thresholds and one outside.
3. Project into multi-spectral representation

Thresholding Along Boundaries

If we want our thresholding method to give stay fairly true to the boundaries of the object, we can first apply some boundary-finding method (such as edge detection techniques) and then sample the pixels only where the boundary probability is high.

Thus, our threshold method based on pixels near boundaries will cause separations of the pixels in ways that tend to preserve the boundaries. Other scattered distributions within the object or the background are of no relevance.

However, if the characteristics change along the boundary, we're still in trouble. And, of course, there's still no guarantee that we'll not have extraneous pixels or holes.

Adaptive Thresholding

- Uneven illumination
  - An image can be modeled as the product of a reflectance component r(x,y) and an illumination component i(x,y) as

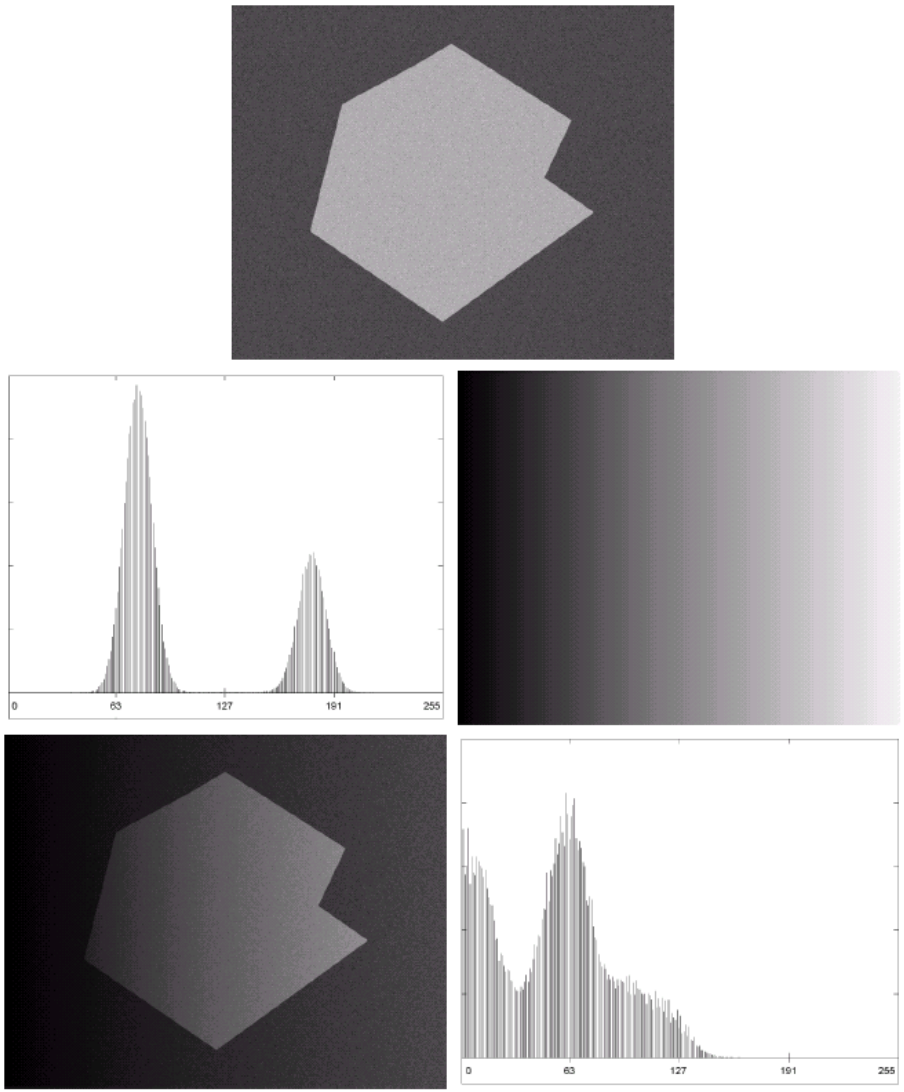
$$f(x,y) = i(x,y)r(x,y)$$

where

$$0 < i(x,y) < \infty \qquad \text{and} \qquad 0 < r(x,y) < 1$$

  - i(x,y): the amount of source illumination incident on the scene being viewed
  - r(x,y): the amount of illumination reflected by the object

Adaptive Thresholding



- Uneven illumination makes an originally perfectly segmentable image into an image that can not be segmented satisfactorily using a single threshold

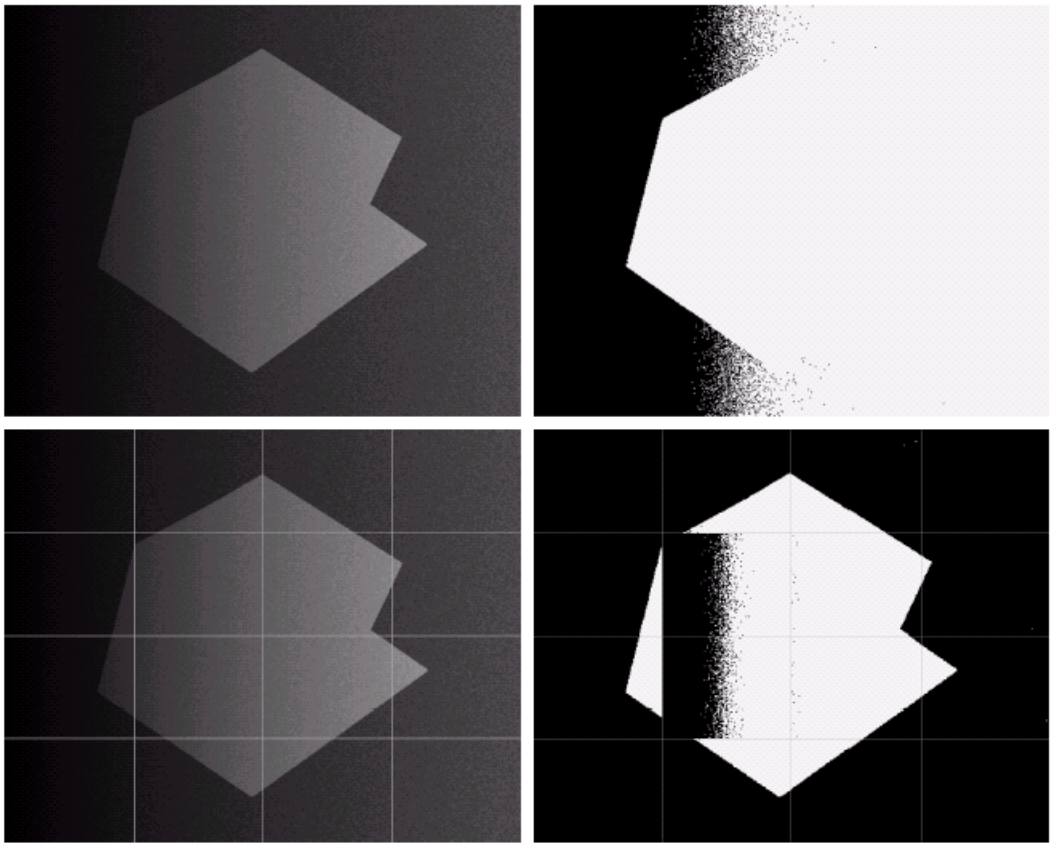
Adaptive Thresholding

- One way to overcome the uneven illumination problem is to first estimate the uneven illumination and then correct it accordingly (rectification)
  - Upon correction, global thresholding can be employed
- Another way is to use adaptive thresholding by partition the original image into several subimages and utilize global thresholding techniques for each subimage
  - Key issues are how to partition the image and how to estimate the threshold for each subimage

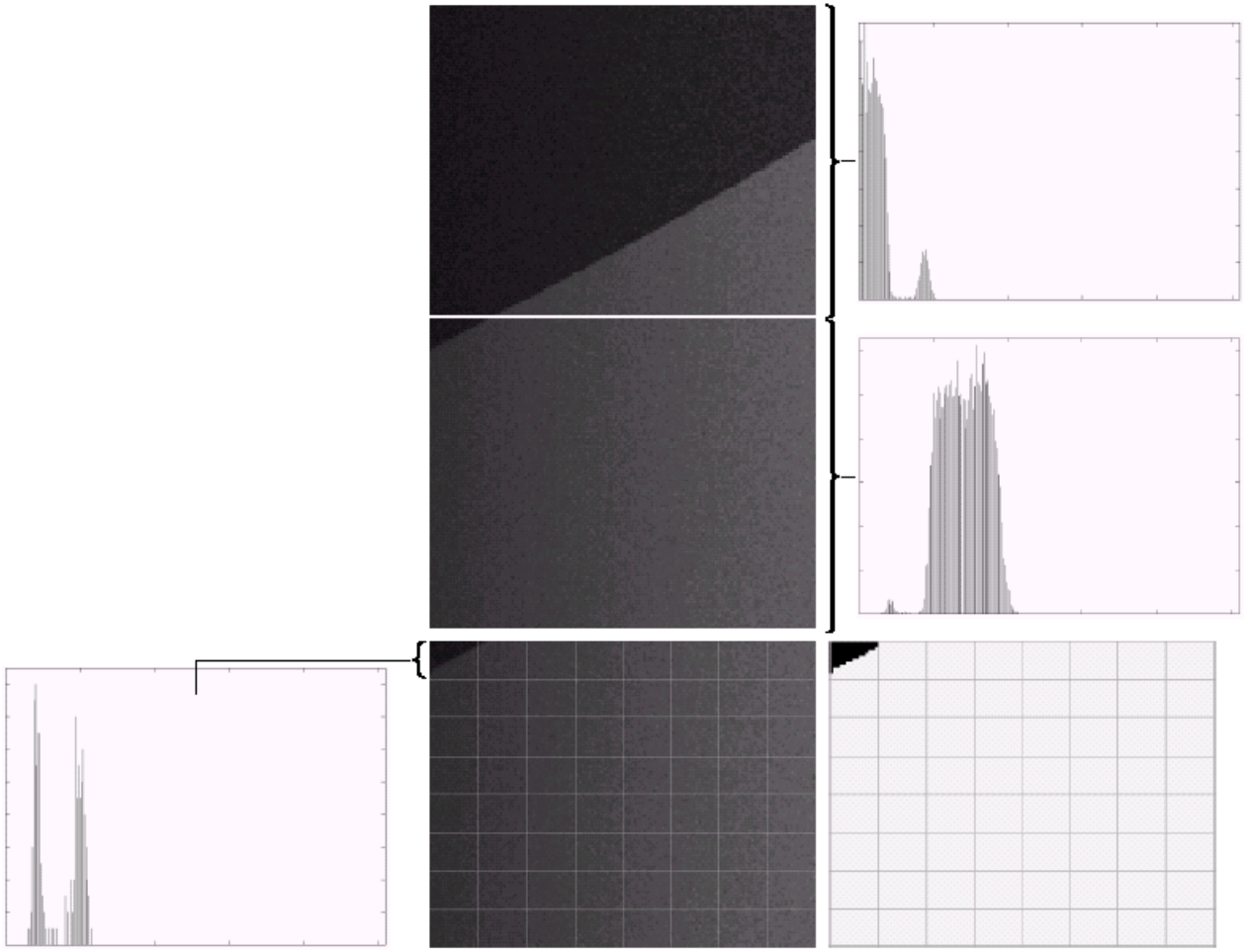
Adaptive Thresholding Examples

a b
c d

FIGURE 10.30
(a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.



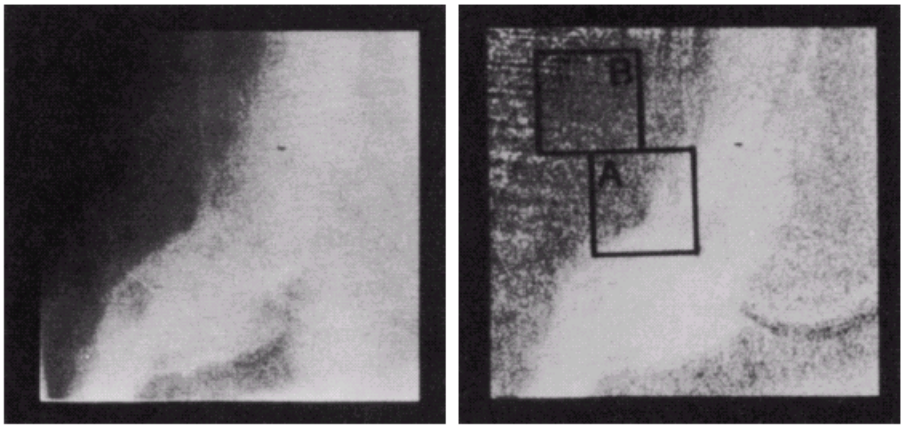
Adaptive Thresholding Examples

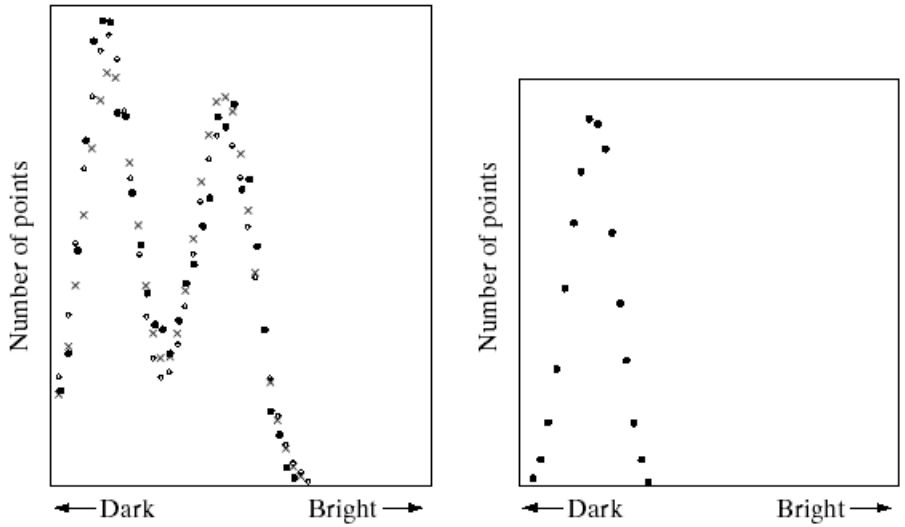


Adaptive Thresholding Examples

a b

FIGURE 10.33 A cardioangiogram before and after preprocessing. (Chow and Kaneko.)

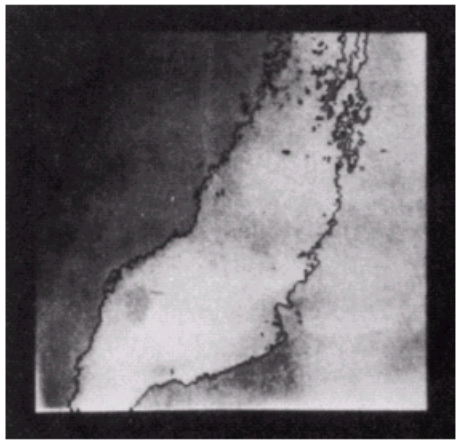




a b

FIGURE 10.34 Histograms (black dots) of (a) region *A*, and (b) region *B* in Fig. 10.33(b). (Chow and Kaneko.)

Adaptive Thresholding Examples



- Major procedures used for this example
  - Dividing the image into subimages
  - Testing for bimodality for each subimage
  - Apply Optimal Global Thresholding for each identified image with bimodal histogram

optimal thresholds can be determined for each voxel and used for segmentation. In [Fra et al. 95, Santago and Gage 93], the partial volume effect was also considered (in brain M images, the finite-size voxels can consist of a combination of, e.g., gray and white matte and a volume percentage corresponding to WM, GM, and CSF was calculated for each voxe Figure 5.6 gives an example of such brain segmentation. The brighter the voxel location individual segmented images, the higher the volume percentage of the GM, WM, or CSF that particular voxel. In each voxel, the sum of partial volume percentages is 100%.

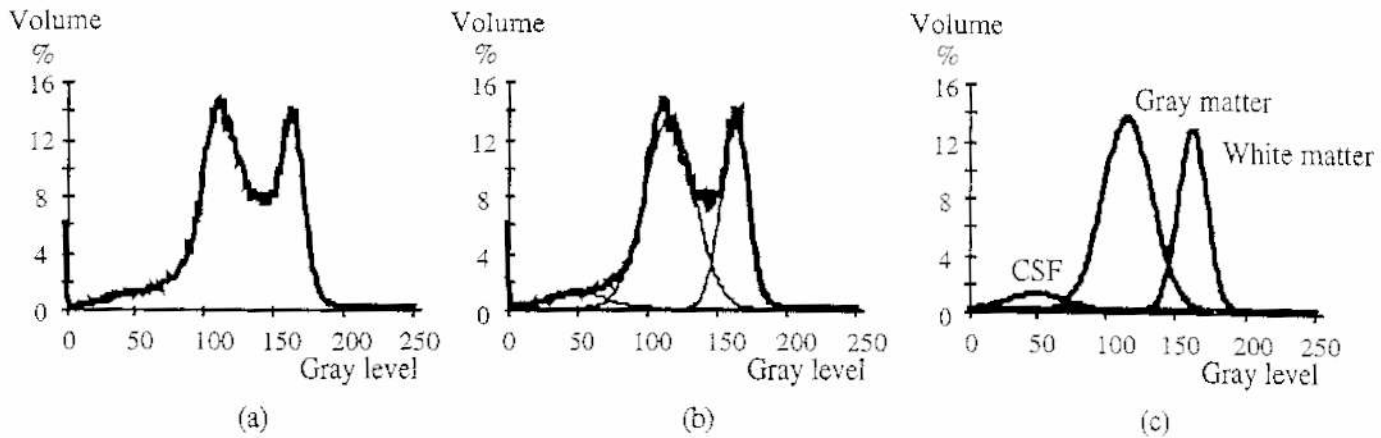


Figure 5.5: *Segmentation of 3D T1-weighted MR brain image data using optimal thresholding: (a) local gray-level histogram; (b) fitted Gaussian distributions, global 3D image fi (c) Gaussian distributions corresponding to WM, GM, and CSF. Courtesy R.J. Frank, T. Grabowski, Human Neuroanatomy and Neuroimaging Laboratory, Department of Neurology The University of Iowa.*

5.1.3 Multi-spectral thresholding

Many practical segmentation problems need more information than is contained in one spe tral band. Color images are a natural example, in which information is coded in three spectra bands, for example, red, green, and blue; multi-spectral remote sensing images or meteoro logical satellite images may have even more spectral bands. One segmentation approac determines thresholds independently in each spectral band and combines them into a sing segmented image.

Algorithm 5.3: Recursive multi-spectral thresholding

1. Initialize the whole image as a single region.
2. Compute a smoothed histogram (see Section 2.3.2) for each spectral band. Find th most significant peak in each histogram and determine two thresholds as local minima o either side of this maximum. Segment each region in each spectral band into sub-region according to these thresholds. Each segmentation in each spectral band is projecte into a multi-spectral segmentation—see Figure 5.7. Regions for the next processin steps are those in the multi-spectral image.
3. Repeat step 2 for each region of the image until each region's histogram contains onl one significant peak.