Chapter 3

Regions

A region in an image is a group of connected pixels with similar properties. Regions are important for the interpretation of an image because they may correspond to objects in a scene. An image may contain several objects and, in turn, each object may contain several regions corresponding to different parts of the object. For an image to be interpreted accurately, it must be partitioned into regions that correspond to objects or parts of an object. However, due to segmentation errors, the correspondence between regions and objects will not be perfect, and object-specific knowledge must be used in later stages for image interpretation.

3.1 Regions and Edges

Consider the simple image shown in Figure 3.1. This figure contains several objects. The first step in the analysis and understanding of this image is to partition the image so that regions representing different objects are explicitly marked. Such partitions may be obtained from the characteristics of the gray values of the pixels in the image. Recall that an image is a two-dimensional array and the values of the array elements are the gray values. Pixels, gray values at specified indices in the image array, are the observations, and all other attributes, such as region membership, must be derived from the gray values. There are two approaches to partitioning an image into regions: region-based segmentation and boundary estimation using edge detection.

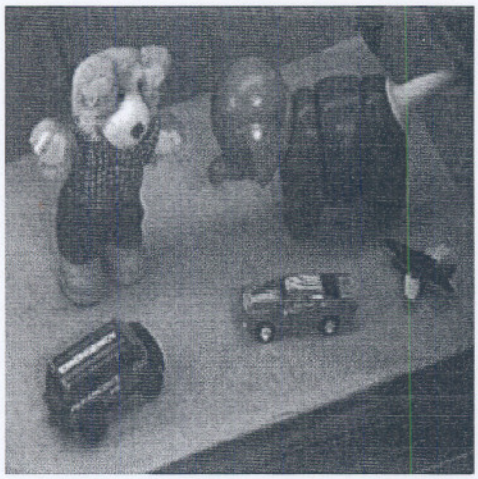


Figure 3.1: This figure shows an image with several regions. Note that regions and boundaries contain the same information because one representation can be derived from the other.

In the region-based approach, all pixels that correspond to an object are grouped together and are marked to indicate that they belong to one region. This process is called *segmentation*. Pixels are assigned to regions using some criterion that distinguishes them from the rest of the image. Two very important principles in segmentation are *value similarity* and *spatial proximity*. Two pixels may be assigned to the same region if they have similar intensity characteristics or if they are close to one another. For example, a specific measure of value similarity between two pixels is the difference between the gray values, and a specific measure of spatial proximity is Euclidean distance. The variance of gray values in a region and the compactness of a region can also be used as measures of value similarity and spatial proximity of pixels within a region, respectively.

The principles of similarity and proximity come from the assumption that points on the same object will project to pixels in the image that are spatially close and have similar gray values. Clearly, this assumption is not satisfied in many situations. We can, however, group pixels in the image using these simple assumptions and then use domain-dependent knowledge to match regions to object models. In simple situations, segmentation can be done with thresholding and component labeling, as discussed in Chapter 2. Complex

images may require more sophisticated techniques than thresholding to assign pixels to regions that correspond to parts of objects.

Segmentation can also be done by finding the pixels that lie on a region boundary. These pixels, called edges, can be found by looking at neighboring pixels. Since edge pixels are on the boundary, and regions on either side of the boundary may have different gray values, a region boundary may be found by measuring the difference between neighboring pixels. Most edge detectors use only intensity characteristics as the basis for edge detection, although derived characteristics, such as texture and motion, may also be used.

In ideal images, a region will be bounded by a closed contour. In principle, region segmentation and edge detection should yield identical results. Edges (closed contours) may be obtained from regions using a boundary-following algorithm. Likewise, regions may be obtained from edges using a region-filling algorithm. Unfortunately, in real images it is rare to obtain correct edges from regions and vice versa. Due to noise and other factors, neither region segmentation nor edge detection provides perfect information.

In this chapter, we will discuss the basic concepts of regions, concentrating on two issues:

- Segmenting an image into regions
- Representing the regions

This chapter begins with a discussion of automatic thresholding and histogram methods for segmentation, followed by a discussion of techniques for representing regions. Then more sophisticated techniques for region segmentation will be presented. Edge detection techniques will be discussed in Chapter 5.

In the following section, we will discuss techniques for region formation. Thresholding is the simplest region segmentation technique. After discussing thresholding, we will present methods to judge the similarity of regions using their intensity characteristics. These methods may be applied after an initial region segmentation using thresholding. It is expected that these algorithms will produce a region segmentation that corresponds to an object or its part. Motion characteristics of points can also be used to form and refine regions. Knowledge-based approaches may be used to match regions to object models. The use of motion will be discussed in Chapter 14.

3.2 Region Segmentation

The segmentation problem, first defined in Section 2.1, is now repeated for ease of reference: Given a set of image pixels $\mathcal{I}$ and a homogeneity predicate $P(\cdot)$, find a partition S of the image $\mathcal{I}$ into a set of n regions R_i,

$$\bigcup_{i=1}^{n} R_i = \mathcal{I}.$$

The homogeneity predicate and partitioning of the image have the properties that any region satisfies the predicate

$$P(R_i) = \text{True}$$

for all i, and any two adjacent regions cannot be merged into a single region that satisfies the predicate

$$P(R_i \cup R_j) = \text{False}.$$

The homogeneity predicate $P(\cdot)$ defines the conformity of all points in the region R_i to the region model.

The process of converting a gray value image into a binary image is a simple form of segmentation where the image is partitioned into two sets. The algorithms for thresholding to obtain binary images can be generalized to more than two levels. The thresholds in the algorithm discussed in Chapter 2 were chosen by the designer of the system. To make segmentation robust to variations in the scene, the algorithm should be able to select an appropriate threshold automatically using the samples of image intensity present in the image. The knowledge about the gray values of objects should not be hard-wired into an algorithm; the algorithm should use knowledge about the relative characteristics of gray values to select the appropriate threshold. This simple idea is useful in many computer vision algorithms.

3.2.1 Automatic Thresholding

To make segmentation more robust, the threshold should be automatically selected by the system. Knowledge about the objects in the scene, the application, and the environment should be used in the segmentation algorithm in a form more general than a fixed threshold value. Such knowledge may include

- Intensity characteristics of objects
- Sizes of the objects
- Fractions of an image occupied by the objects
- Number of different types of objects appearing in an image

A thresholding scheme that uses such knowledge and selects a proper threshold value for each image without human intervention is called an automatic thresholding scheme. Automatic thresholding analyzes the gray value distribution in an image, usually by using a histogram of the gray values, and uses the knowledge about the application to select the most appropriate threshold. Since the knowledge employed in these schemes is more general, the domain of applicability of the algorithm is increased.

Suppose that an image contains n objects $O_1, O_2, \ldots, O_n$, including the background, and gray values from different populations $\pi_1, \ldots, \pi_n$ with probability distributions $p_1(z), \ldots, p_n(z)$. In many applications, the probabilities $P_1, \ldots, P_n$ of the objects appearing in an image may also be known. Using this knowledge, it is possible to rigorously formulate the threshold selection problem. Since the illumination geometry of a scene controls the probability distribution of intensity values $p_i(z)$ in an image, one cannot usually precompute the threshold values. As we will see, most methods for automatic threshold selection use the size and probability of occurrence and estimate intensity distributions by computing histograms of the image intensities.

Many automatic thresholding schemes have been used in different applications. Some of the common approaches are discussed in the following sections. To simplify the presentation, we will follow the convention that objects are dark against a light background. In discussing thresholds, this allows us to say that gray values below a certain threshold belong to the object and gray values above the threshold are from the background, without resorting to more cumbersome language. The algorithms that we present in the following sections can easily be modified to handle other cases such as light objects against a dark background, medium gray objects with background values that are light and dark, or objects with both light and dark gray values against a medium gray background. Some algorithms can be generalized to handle object gray values from an arbitrary set of pixel values.

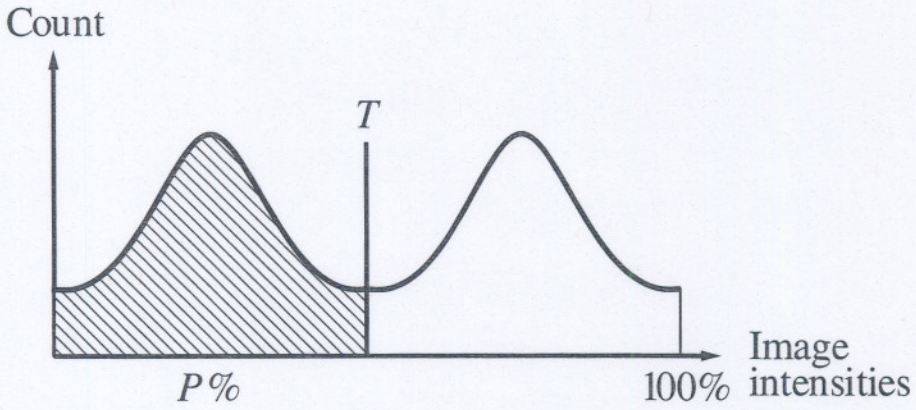


Figure 3.2: The shaded areas in the histogram represent p percent of the image area. The threshold is selected so that p percent of the histogram is assigned to the object.

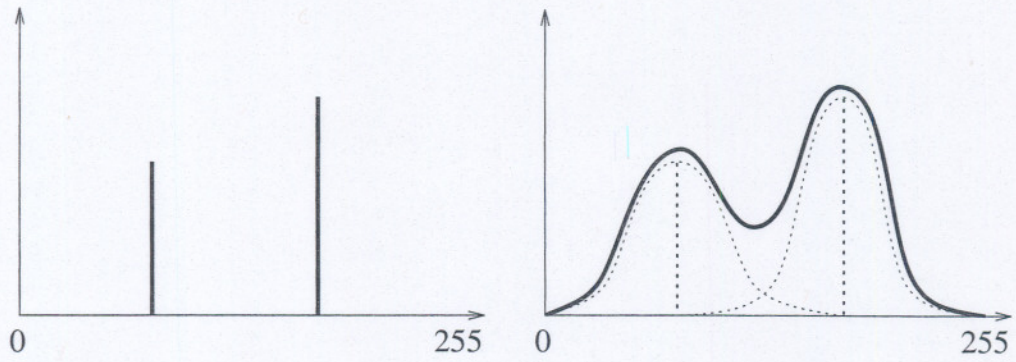
P-Tile Method

The p-tile method uses knowledge about the area or size of the desired object to threshold an image. Suppose that in a given application objects occupy about p percent of the image area. By using this knowledge to partition the gray value histogram of the input image, one or more thresholds can be chosen that assign p percent of the pixels to the object. Figure 3.2 gives an example of a binary image formed using this technique.

Clearly, this method is of very limited use. Only a few applications, such as page readers, allow such an estimate of the area in a general case.

Mode Method

If the objects in an image have the same gray value, the background has a different gray value, and the image pixels are affected by zero-mean Gaussian noise, then we may assume that the gray values are drawn from two normal distributions with parameters (μ_1, σ_1) and (μ_2, σ_2). The histogram for an image will then show two separate peaks, as shown in Figure 3.3. In the ideal case of constant intensity values, $\sigma_1 = \sigma_2 = 0$, there will be two spikes in the histogram and the threshold can be placed anywhere between the spikes. In practice, the two peaks are not so well separated. In this case, we may detect



(a) (b)

Figure 3.3: Ideally, the intensities of the background and objects will be widely separated. In the ideal case, the threshold T can be anywhere between the two peaks, as shown in (a). In most images, the intensities will overlap, resulting in histograms as shown in (b).

peaks and valleys in the histogram, and the threshold may be set to the pixel value corresponding to the valley. It can be shown that the probability of misclassification is minimized by this choice of the threshold when the size of the object is equal to that of the background (see Exercise 3.2). In most applications, since the histogram is sparsely populated near the valley, the segmentation is not sensitive to the threshold value.

The determination of peaks and valleys is a nontrivial problem, and many methods have been proposed to solve it. For an automatic thresholding scheme, we should have a measure of the *peakiness* and *valleyness* of a point in a histogram. A computationally efficient method is given in Algorithm 3.1. This method ignores local peaks by considering peaks that are at some minimum distance apart. The peakiness is based on the height of the peaks and the depth of the valleys; the distance between the peaks and valleys is ignored.

This approach can be generalized to images containing many objects with different mean gray values. Suppose there are n objects with normally distributed gray values with parameters $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_n, \sigma_n)$, and the background is also normally distributed with parameters (μ_0, σ_0). If the means are significantly different, the variances are small, and none of the ob-

Algorithm 3.1 Peakiness Detection for Appropriate Threshold Selection

1. *Find the two highest local maxima in the histogram that are at some minimum distance apart. Suppose these occur at gray values* g_i *and* g_j.
2. *Find the lowest point* g_k *in the histogram* H *between* g_i *and* g_j.
3. *Find the peakiness, defined as* $\min(H(g_i), H(g_j))/H(g_k)$.[1]
4. *Use the combination* (g_i, g_j, g_k) *with highest peakiness to threshold the image. The value* g_k *is a good threshold to separate objects corresponding to* g_i *and* g_j.

jects is very small in size, then the histogram for the image will contain $n+1$ peaks. The valley locations $T_1, T_2, \ldots, T_n$ can be determined, and pixels with gray values in each interval $(T_i, T_{i+1}]$ can be assigned to the corresponding object (see Figure 3.4).

Iterative Threshold Selection

An iterative threshold selection method starts with an approximate threshold and then successively refines this estimate. It is expected that some property of subimages resulting from the threshold can be used to select a new threshold value that will partition the image better than the first threshold. The threshold modification scheme is critical to the success of this approach. The method is given in Algorithm 3.2.

Adaptive Thresholding

If the illumination in a scene is uneven, then the above automatic thresholding schemes may not be suitable. The uneven illumination may be due to shadows or due to the direction of illumination. In all such cases, the same threshold value may not be usable throughout the complete image (see Figure 3.5). *Non-adaptive* methods analyze the histogram of the entire image.

[1] If the valley region shows a great deal of "spikiness" with many empty bins, then a certain amount of smoothing may be required to remove the possibility of division by zero.

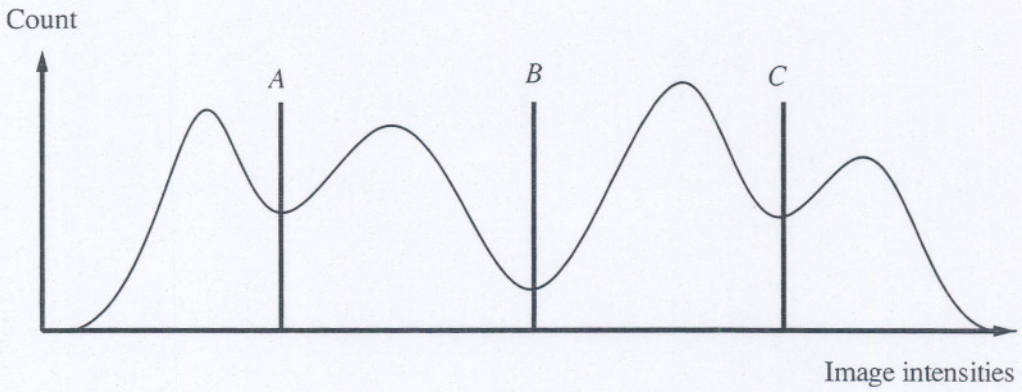


Figure 3.4: Histogram for an image containing several objects with differing intensity values.

Methods to deal with uneven illumination or uneven distribution of gray values in the background should look at a small region of an image and then analyze this subimage to obtain a threshold for only that subimage. Some techniques have been developed to deal with this kind of situation.

A straightforward approach to segment such images is to partition the image into $m \times m$ subimages and select a threshold T_{ij} for each subimage based on the histogram of the ijth subimage ($1 \leq i, j \leq m$). The final segmentation of the image is the union of the regions of its subimages. The results of this method are shown in Figure 3.6.

Variable Thresholding

Another useful technique in the case of uneven illumination is to approximate the intensity values of the image by a simple function such as a plane or biquadratic. The function fit is determined in large part by the gray value of the background. Histogramming and thresholding can be done relative to the base level determined by the fitted function. This technique is also called background normalization. For example, Figure 3.7 shows a three-dimensional plot of the box image with uneven illumination. If a planar function were fitted to this function, it would lay somewhere between the two rough surfaces representing the box and the background. Now, by using this fitted plane as the basis for thresholding, the box can be easily segmented. Any points in the image that are above the plane will be part of the box, and anything below will be part of the background.

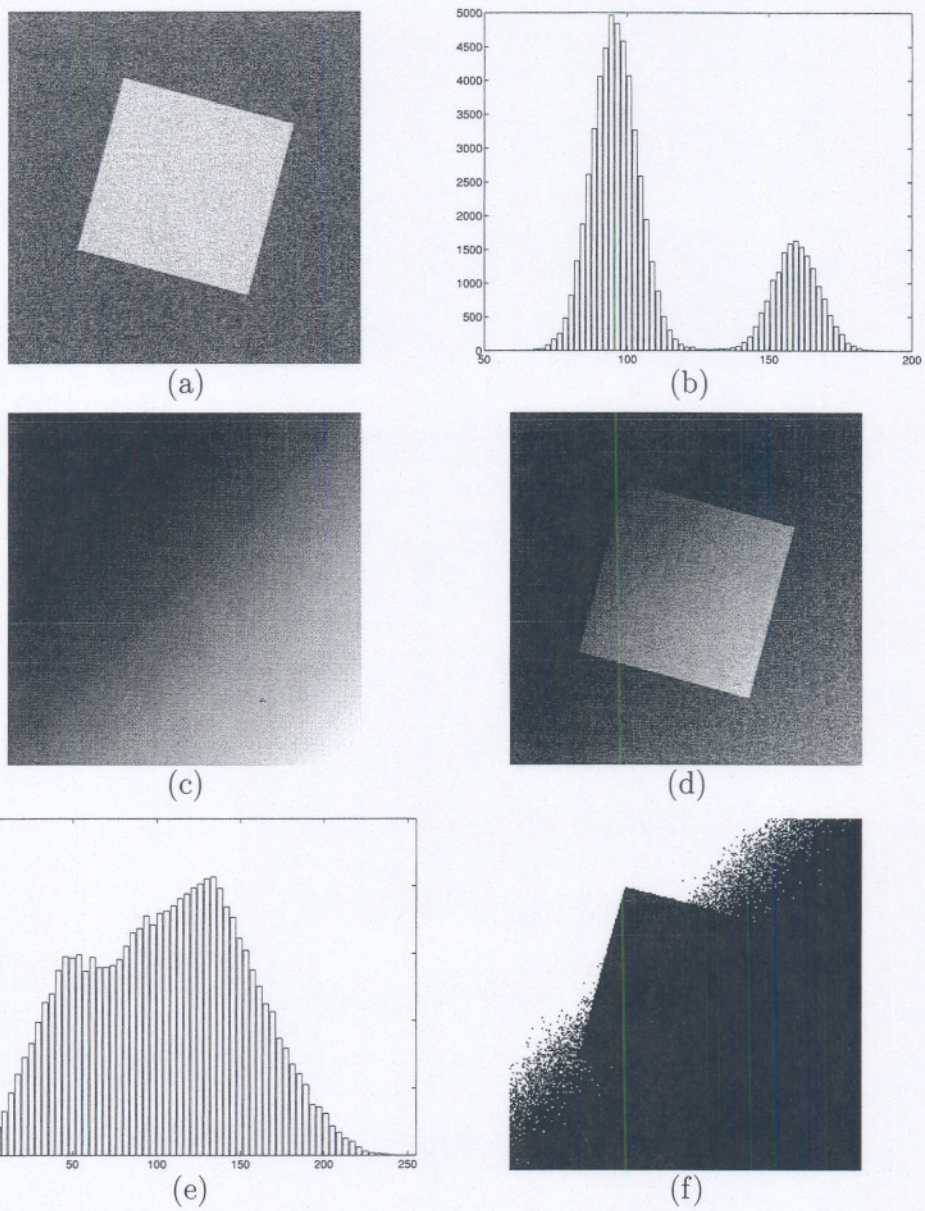


Figure 3.5: An example of an image with uneven illumination which is not amenable to regular thresholding. (a) Original image with uniform illumination. (b) Histogram of original. (c) Simulated uneven illumination. (d) Box with uneven illumination. (e) Histogram of box with uneven illumination. (f) Box thresholded at approximate valley of histogram, $T = 72$. Note that regular thresholding is not effective in segmenting the object from the background.

Algorithm 3.2 Iterative Threshold Selection

1. *Select an initial estimate of the threshold, T. A good initial value is the average intensity of the image.*
2. *Partition the image into two groups, R_1 and R_2, using the threshold T.*
3. *Calculate the mean gray values μ_1 and μ_2 of the partitions R_1 and R_2.*
4. *Select a new threshold:*

$$T = \frac{1}{2}(\mu_1 + \mu_2).$$

5. *Repeat steps 2–4 until the mean values μ_1 and μ_2 in successive iterations do not change.*

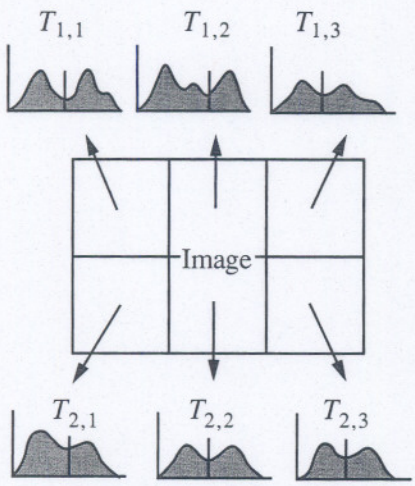


Figure 3.6: The process of adaptive thresholding on an image which is not amenable to regular thresholding.

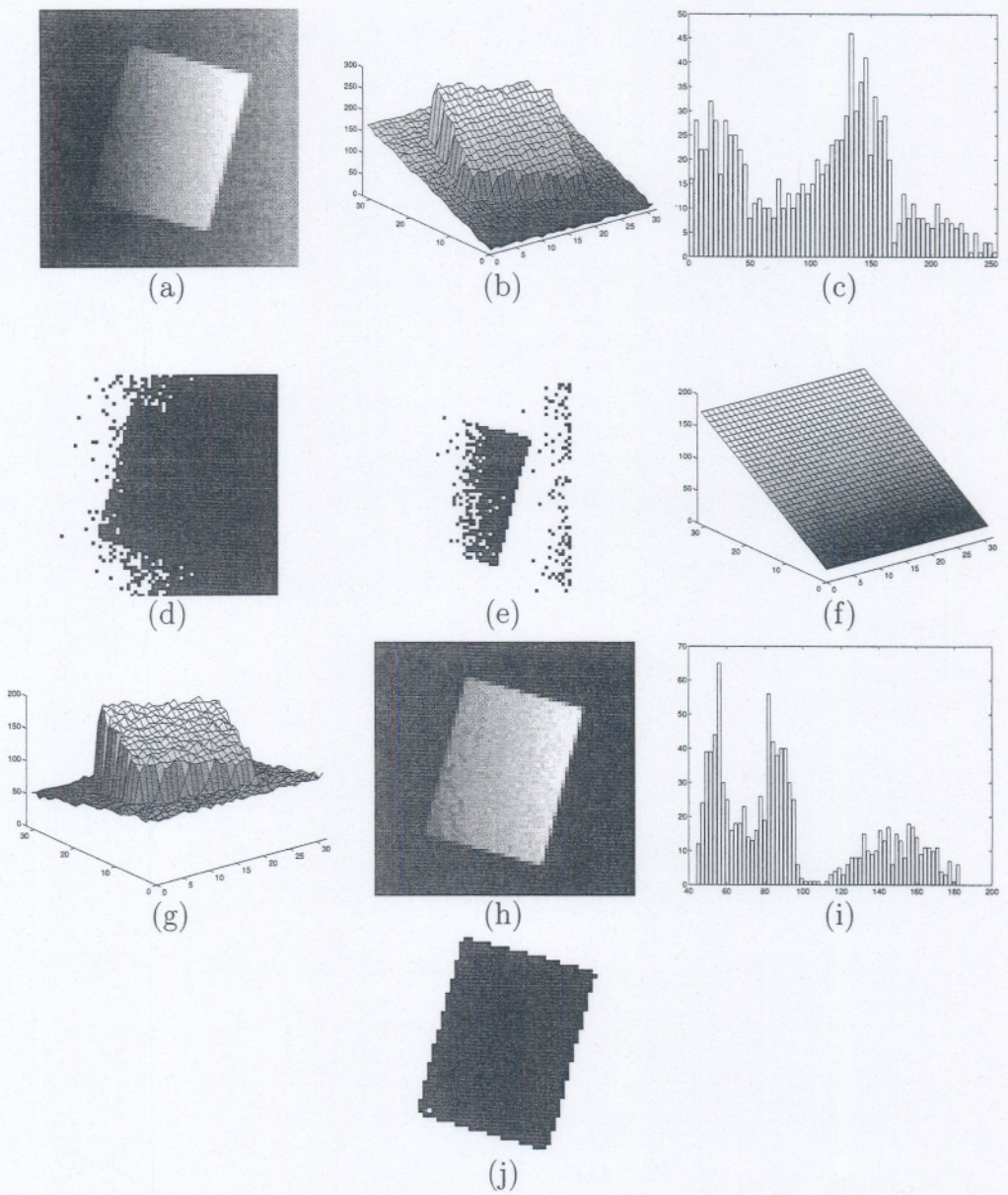


Figure 3.7: The results of variable thresholding on an image that is not amenable to regular thresholding. (a) Original image with uneven illumination. (b) 3-D plot of original image. (c) Histogram of original image. (d) Thresholded original, $T = 85$. (e) Thresholded original, $T = 165$. (f) Approximated planar function. (g) Plot of difference between original image and fitted plane. This is the *normalized* image. (h) The normalized image. (i) Histogram of normalized image. (j) Thresholded result, $T = 110$.

Algorithm 3.3 Double Thresholding for Region Growing

1. *Select two thresholds T_1 and T_2.*
2. *Partition the image into three regions: R_1, containing all pixels with gray values below T_1; R_2, containing all pixels with gray values between T_1 and T_2, inclusive; and R_3, containing all pixels with gray values above T_2.*
3. *Visit each pixel assigned to region R_2. If the pixel has a neighbor in region R_1, then reassign the pixel to region R_1.*
4. *Repeat step 3 until no pixels are reassigned.*
5. *Reassign any pixels left in region R_2 to region R_3.*

Double Thresholding

In many applications, it is known that certain gray values belong to objects. However, there may be additional gray values that belong to either objects or the background. In such a case, one may use a conservative threshold T_1 to obtain the *core* of the object and then use some method to *grow* the object regions. The methods used for growing these regions will depend on the specific application. Common approaches include using another threshold to accept pixels if they have a neighbor that is a core pixel or by using intensity characteristics, such as a histogram, to determine points to be included in the object region. A simple approach is to accept all points that are below a second threshold T_2 and are connected to the original set of points. This approach is outlined in Algorithm 3.3, and the results are shown in Figure 3.8.

In Algorithm 3.3, region R_1 is the core region, region R_2 is the fringe region (also called the intermediate or transition region), and region R_3 is the background. The core region is grown by adding pixels from the fringe that are neighbors of core pixels. After region growing, any pixels that are not in the core region are background pixels. The double thresholding algorithm for region growing implements the principles of value similarity and spatial proximity. Fringe pixels have values that are close to the values of core pixels

since the two sets of pixels are adjacent in the histogram, and fringe pixels are spatially close to core pixels since they are neighbors.

3.2.2 Limitations of Histogram Methods

As discussed above, one may use information in the histogram of an image to select an appropriate threshold for segmentation. This approach is useful in those applications where objects have constant gray values. If the illumination is different in different parts of a scene, then a single threshold may not be sufficient to segment the image, even if the image contains only one object. In such cases one must use techniques that effectively partition an image, arbitrarily, and select thresholds for each subimage independently. We also saw some other heuristics for using histogram-based segmentation. If the images are complex, these approaches will also perform poorly.

The most basic limitation of the histogram-based approaches is due to the fact that a histogram throws away spatial information about the intensity values in an image. The histogram describes the global intensity distribution. Several images with very different spatial distributions of gray values may have similar histograms. For example, one cannot distinguish between a random distribution of black and white points, a binary checkerboard, and an image that is half black and half white just on the basis of their histograms. The global nature of a histogram limits its applicability to complex scenes. It does not exploit the important fact that points from the same object are usually spatially close due to *surface coherence*.

3.3 Region Representation

Regions are used in many contexts and can be represented in many alternative forms. Different representations are suitable in different applications. Some applications require computations only for a single region, while others require relationships among different regions of an image. In this section, we will discuss a few commonly used representations of regions and study their features. It must be mentioned here that regions can also be represented as closed contours. We will discuss those representations separately in Chapter 6.

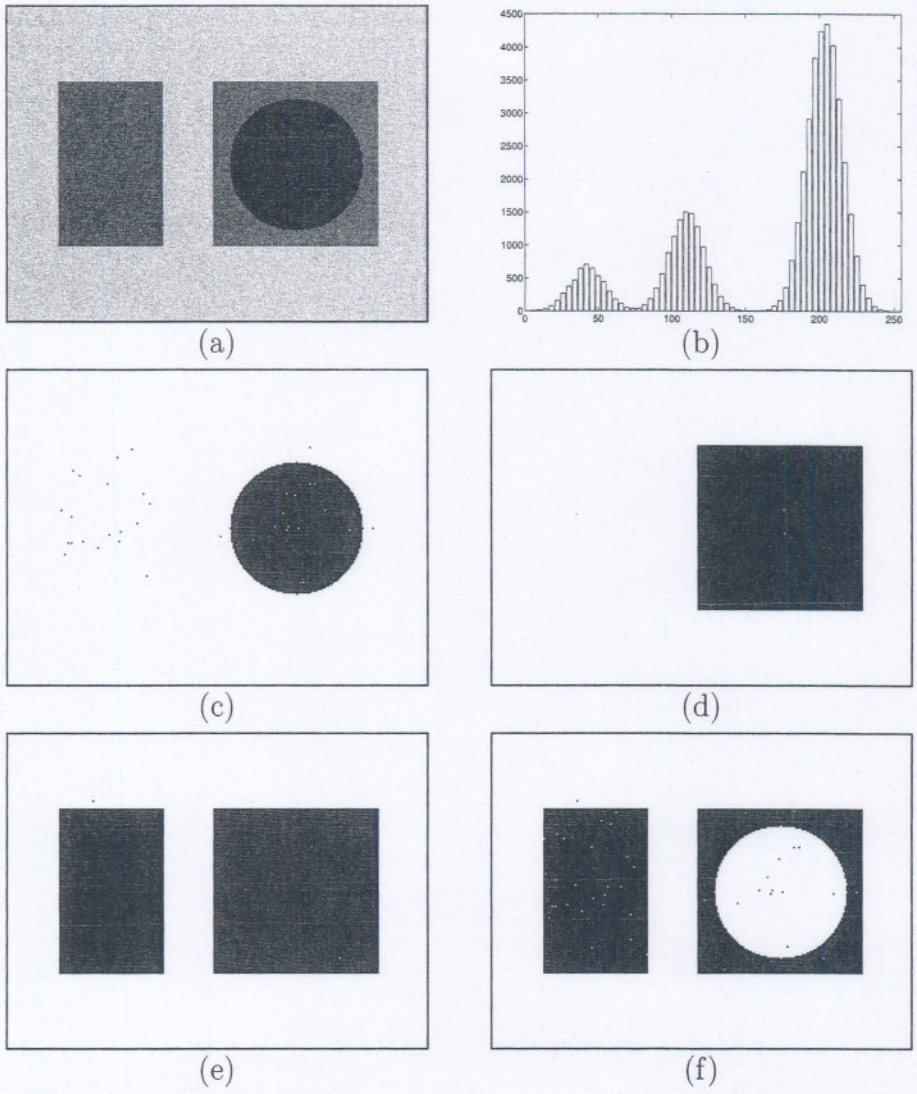


Figure 3.8: An image that is difficult to segment using a single threshold and its segmentation using double thresholding. (a) Original image. (b) Histogram of original image. (c) Core region, $T_1 = 70$. (d) Results after growing the core region using a second threshold, $T_2 = 155$. Compare this with the results using regular thresholding shown in (e) and (f). (e) Threshold $= T_2$. (f) $T_1 \leq$ Threshold $\leq T_2$. Note that it is impossible to segment the entire right square without using region growing threshold.

Most region representations can be classified into one of the following three classes:

1. Array representations
2. Hierarchical representations
3. Symbolic representations

3.3.1 Array Representation

The basic representation for regions is to use an array of the same size as the original image with entries that indicate the region to which a pixel belongs. Referring back to Chapter 2, Figure 2.12(b) is a representation of regions in the image shown in Figure 2.12(a). Thus, if element i, j of the array has the value α, then the corresponding pixel in the image belongs to region α. The simplest example of this representation is a binary image where each pixel belongs to either region 0, the background, or region 1, the foreground.

Another scheme uses membership arrays (images), commonly called masks or bitmaps. Each region is associated with a binary image, its mask, that indicates which pixels belong to that region. By overlaying masks on the original image, intensity characteristics of regions can be found. One advantage to this scheme is that ambiguous situations, where the region membership of a pixel cannot be definitely decided, can be handled by allowing the pixel to be a member of more than one region. The corresponding pixel will be 1 in more than one mask. Array representations contain region information in iconic or image form. Symbolic information is not explicitly represented.

The last few years have seen increased use of gray value images in computer applications. This popularity is due to widespread use of raster graphics terminals and decreased memory costs. Array representation preserves all details of regions required in most applications. This has made binary masks a very popular representation in computers and much hardware support is available to manipulate them.

3.3.2 Hierarchical Representations

Images can be represented at many different resolutions. Clearly, by reducing an image's resolution, thus reducing the size of the array, some data is

lost, making it more difficult to recover information. However, reduction in resolution results in reduced memory and computing requirements. Hierarchical representation of images allows representation at multiple resolutions. In many applications, one can compute properties of images first at a low resolution and then perform additional computations over a selected area of the image at a higher resolution. Hierarchical representations are also used for browsing in images. We present two commonly used forms of hierarchical representations: pyramids and quad trees.

Pyramids

A pyramid representation of an $n \times n$ image contains the image and k reduced versions of the image. Usually n is a power of 2 and the other images are $n/2 \times n/2$, $n/4 \times n/4, \ldots, 1 \times 1$. In a pyramid representation of an image, the pixel at level l is obtained by combining information from several pixels in the image at level $l+1$. The whole image is represented as a single pixel at the top level, level 0, and the bottom level is the original (unreduced) image. A pixel at a level represents aggregate information represented by several pixels at the next level. Figure 3.9 shows an image and its reduced versions in a pyramid. Here the pyramid is obtained by simply averaging the gray values in 2×2 neighborhoods. It is possible, however, to devise other strategies to form reduced-resolution versions. Similarly, it is possible to taper the pyramid in nonlinear ways.

An implementational point is that the entire pyramid fits into a linear array of size $2(2^{2\times \text{level}})$.

Quad Trees

A quad tree may be considered an extension of pyramids for binary images. A quad tree contains three types of nodes: white, black, and gray. A quad tree is obtained by recursive splitting of an image. A region in an image is split into four subregions of identical size, as shown in Figure 3.10. For each subregion, if all points in the region are either white or black, then this region is no longer considered as a candidate for splitting; if it contains pixels of both kinds, it is considered to be a "gray region" and is further split into four subregions. An image obtained using this recursive splitting is represented in a tree structure. The splitting process is repeated until there are no gray

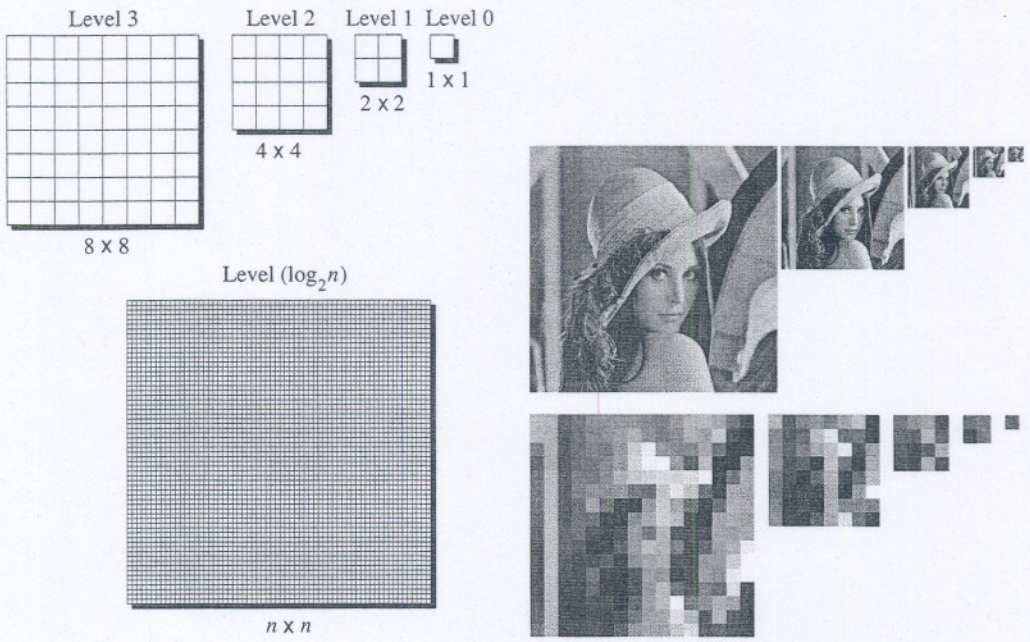


Figure 3.9: The original image is a 512×512 image; its reduced-resolution versions are successively obtained by averaging four points. All successive versions are shown here. Note that the low resolution images have been enlarged for display.

regions in the tree. Each node in this structure is either a leaf node or has four children—thus the name *quad tree.*

Quad trees are finding increasing application in spatial databases. Several algorithms have been developed for converting a raster array to a quad tree and a quad tree to a raster array. Algorithms for computing several pictorial properties have also been developed. The last few years have seen efforts to represent a quad tree using codes to reduce the memory required by pointers.

3.3.3 Symbolic Representations

A region can be represented using symbolic characteristics. Some commonly used symbolic characteristics are:

- Enclosing rectangle

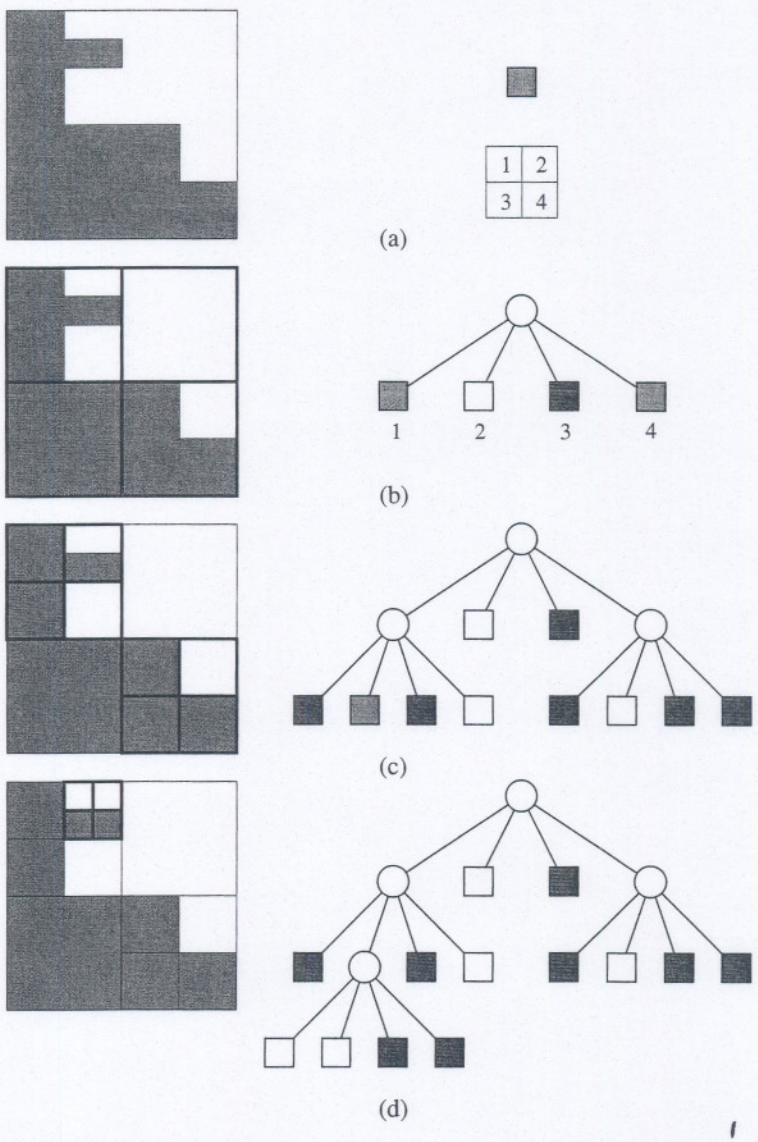


Figure 3.10: The building of a quad tree. (a) Original image, "gray region." (b) Original split into four subregions (the left node in the tree corresponds to the top left region in the image). Note that two of these regions are also gray regions. (c) Splitting the gray regions from (b) into four subregions. One of these regions is still a gray region. (d) Splitting of the last gray region and the final quad tree.

- Centroid
- Moments
- Euler number

Other characteristics such as the mean and variance of intensity values, although not symbolic representations, are also commonly used to represent regions in an image. These measures are straightforward generalizations of the calculations introduced in Chapter 2. In addition, other application-dependent features of a region may be represented. When representing an image for interpretation, we may be required to represent relationships among neighboring regions also.

3.3.4 Data Structures for Segmentation

When implementing region merging and splitting algorithms for segmenting an image (discussed in Section 3.4), information about the regions which have been formed must be maintained in some data structure. Since merge and split operations use information about boundaries between regions, as well as general characteristics of the regions, several data structures have been proposed to allow easy manipulation of region characteristics. In this section, we will discuss a few data structures that facilitate region merging and splitting.

Region Adjacency Graphs

A region adjacency graph (RAG) is used to represent regions and relationships among them in an image. The emphasis is on the partitions of an image in the form of regions and the characteristics of each partition. As shown in Figure 3.11, the following conventions are used. Nodes are used to represent regions, and arcs between nodes represent a common boundary between regions. Different properties of regions may be stored in the node data structure. The RAG emphasizes the adjacency of regions and plays a vital role in segmentation. After an initial segmentation based on primitive characteristics such as intensity values, the results may be represented in a RAG, and then regions may be combined to obtain a better segmentation. A method to generate the RAG is given in Algorithm 3.4.

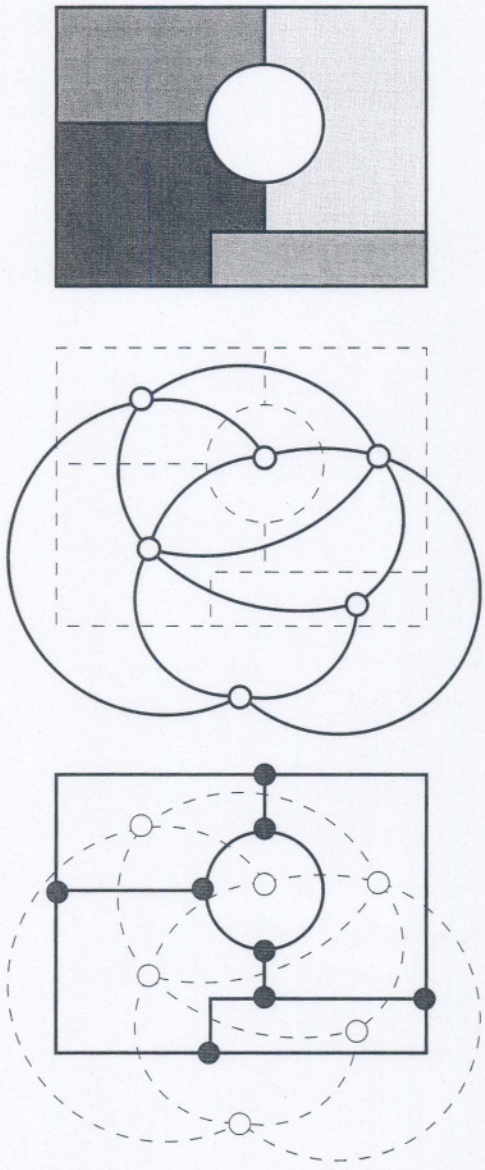


Figure 3.11: The region adjacency graph for a segmented image. *Top:* Segmented image. *Middle:* Region adjacency graph. *Bottom:* The dual of the region agacency graph.

Algorithm 3.4 Region Adjacency Graph

1. *Scan the membership array a and perform the following steps at each pixel index* $[i, j]$.
2. *Let* $r_1 = a[i, j]$.
3. *Visit the neighbors* $[k, l]$ *of the pixel at* $[i, j]$. *For each neighbor, perform the following step.*
4. *Let* $r_2 = a[k, l]$. *If* $r_1 \neq r_2$, *add an arc between nodes* r_1 *and* r_2 *in the region adjacency graph.*

The dual of a region adjacency graph may also be used in some situations. In the dual representation, nodes represent boundaries and arcs represent the regions that are separated by the boundaries.

Picture Trees

The picture tree emphasizes the inclusion of a region within another region. This representation, shown in Figure 3.12, is usually recursive. A version of this representation, the quad tree discussed in an earlier section, has received significant attention. In a picture tree, the emphasis is on nesting regions, while in a quad tree, rectangular regions are split into four rectangles of equal size, independent of the location of regions. A picture tree is usually produced by recursively splitting an image into component parts. Splitting stops when a region with constant characteristics has been reached. This representation will be discussed in detail in a later section on shape representations.

Super Grid

In some applications, it is desirable to store segmentation information in an image array. The representation of boundaries in this situation creates some problems. Intuitively, boundaries should be located *between* the pixels of two adjacent regions. However, often boundaries become actual pixels in image representations. This dilemma is solved by introducing a super grid on the image grid (see Figure 3.13). If the original image is $N \times N$, then the

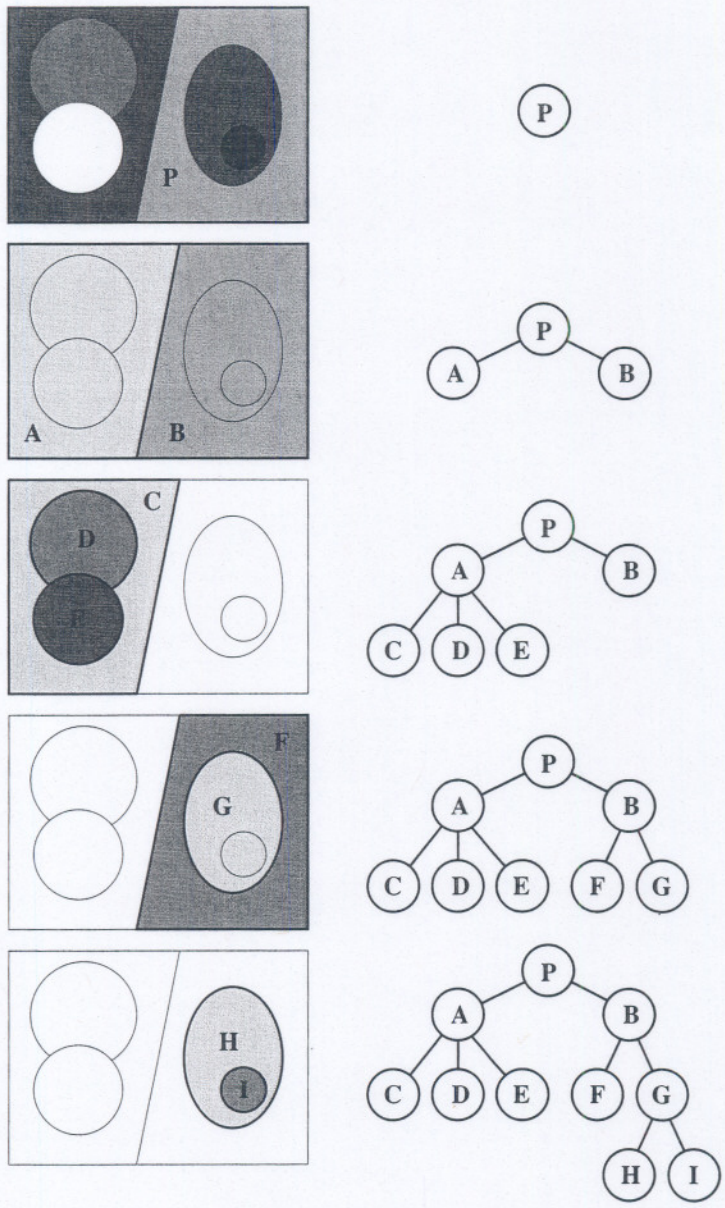


Figure 3.12: An example of the construction of a picture tree representing the inclusion relationships among regions.

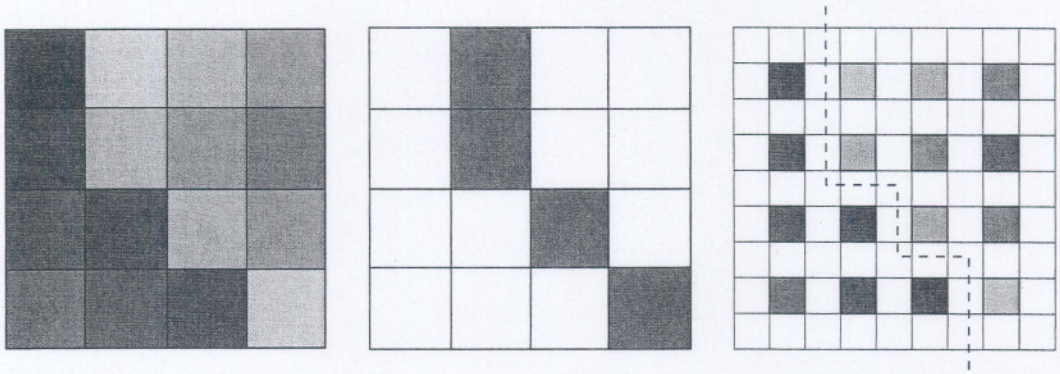


Figure 3.13: A super grid region representation. *Left:* Picture grid. *Middle:* Traditional boundary representation. *Right:* Super grid representation. Note that the boundary is now *between* the two regions, unlike the traditional representation.

supergrid is $(2N+1) \times (2N+1)$. Each pixel is surrounded by eight nonpixel points on the super grid. Nonpixel points are used to indicate whether or not there is a boundary between two pixels, and in what direction the boundary runs. This representation simplifies merge and split operations by explicitly representing the boundary between any two pixels of an image.

3.4 Split and Merge

A simple intensity-based segmentation usually results in too many regions. Even in images where most humans see very clear regions with constant gray value, the output of a thresholding algorithm may contain many extra regions. The main reasons for this problem are high-frequency noise and a gradual transition between gray values in different regions.

After the initial intensity-based region segmentation, the regions may need to be refined or reformed. Several approaches have been proposed for postprocessing such regions obtained from a simple segmentation approach. Some of these approaches use domain-dependent knowledge, while other approaches use knowledge about the imaging process. The refinement may be done interactively by a person or automatically by a computer. In an automatic system, the segmentation will have to be refined based on object

characteristics and general knowledge about the images.

Automatic refinement is done using a combination of split and merge operations. Split and merge operations eliminate false boundaries and spurious regions by merging adjacent regions that belong to the same object, and they add missing boundaries by splitting regions that contain parts of different objects. Some possible approaches for refinement include:

- Merge adjacent regions with similar characteristics.
- Remove questionable edges.
- Use topological properties of the regions.
- Use shape information about objects in the scene.
- Use semantic information about the scene.

The first three approaches use only information about image intensity combined with other domain-independent characteristics of regions. A discussion of approaches for region refinement follows.

3.4.1 Region Merging

The merge operation combines regions that are considered similar. A high-level merge algorithm is given in Algorithm 3.5. The algorithm can be adapted to any of the measures of region similarity discussed in the following sections.

However, when applying a simple algorithm such as this, one may still get into trouble. Consider the following example. We have an image with three adjacent regions A, B, and C. The similarity predicate determines separately that A and B are similar and that B and C are similar. However, A and C are *not*. When merging similar regions, local decisions to merge A and B, and separately B and C, will collapse the three regions into a single region even when A and C are not similar. In this case, one must take additional region characteristics into consideration before merging similar regions.

The most important operation in the merge algorithm is to determine the similarity between two regions. Many approaches have been proposed to judge the similarity of regions. Broadly, the approaches to judge the similarity are based either on the gray value of regions or on the weakness

Algorithm 3.5 Region Merging

1. *Form initial regions in the image using thresholding (or a similar approach) followed by component labeling.*
2. *Prepare a region adjacency graph (RAG) for the image.*
3. *For each region in an image, perform the following steps:*
   (a) *Consider its adjacent region and test to see if they are similar.*
   (b) *For regions that are similar, merge them and modify the RAG.*
4. *Repeat step 3 until no regions are merged.*

of boundaries between the regions, and may include the spatial proximity of regions.

Two approaches to judging the similarity of adjacent regions are:

1. Compare their mean intensities. If the mean intensities do not differ by more than some predetermined value, the regions are considered similar and should be candidates for merging. A modified form of this approach uses surface fitting to determine whether the regions may be approximated by one surface.
2. Assume that the intensity values are drawn from a probability distribution. Consider whether or not to merge adjacent regions based on the probability that they will have the same statistical distribution of intensity values. This approach uses hypothesis testing to judge the similarity of adjacent regions and is discussed in more detail below.

Merging Statistically Similar Regions

This approach considers statistical characteristics of two adjacent regions to decide whether or not they should be merged. Assume that the regions in an image have constant gray value corrupted by statistically independent,

additive, zero-mean Gaussian noise, so that the gray values are drawn from normal distributions. Suppose that two adjacent regions R_1 and R_2 contain m_1 and m_2 points, respectively. There are two possible hypotheses:

H_0: Both regions belong to the same object. In this case, the intensities are all drawn from a single Gaussian distribution with parameters (μ_0, σ_0^2).

H_1: The regions belong to different objects. In this case, the intensities of each region are drawn from separate Gaussian distributions with parameters (μ_1, σ_1^2) and (μ_2, σ_2^2).

In general, these parameters are not known but are estimated using the samples. For example, when a region contains n pixels having gray levels, g_i, $i = 1, 2, \ldots, n$, drawn from a normal distribution given by

$$p(g_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(g_i-\mu)^2}{2\sigma^2}} \tag{3.1}$$

the Maximum Likelihood estimation equations for the parameters are given by

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} g_i \tag{3.2}$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(g_i - \hat{\mu})^2 \tag{3.3}$$

Under the hypothesis H_0, all pixels are independently drawn from a single distribution, $N(\mu_0, \sigma_0^2)$. The joint probability density under H_0 is given by

$$p(g_1, g_2, \ldots, g_{m_1+m_2} \mid H_0) = \prod_{i=1}^{m_1+m_2} p(g_i \mid H_0) \tag{3.4}$$

$$= \prod_{i=1}^{m_1+m_2} \frac{1}{\sqrt{2\pi}\sigma_0} e^{-\frac{(g_i-\mu_0)^2}{2\sigma_0^2}} \tag{3.5}$$

$$= \frac{1}{(\sqrt{2\pi}\sigma_0)^{m_1+m_2}} e^{-\frac{\sum_{i=1}^{m_1+m_2}(g_i-\mu_0)^2}{2\sigma_0^2}} \tag{3.6}$$

$$= \frac{1}{(\sqrt{2\pi}\sigma_0)^{m_1+m_2}} e^{-\frac{(m_1+m_2)}{2}} \tag{3.7}$$

Under the hypothesis H_1, m_1 pixels belong to region 1 with a distribution $N(\mu_1, \sigma_1)$ and m_2 pixels belong to region 2 with a distribution $N(\mu_2, \sigma_2^2)$. Under this hypothesis, the joint density function is given by

$$p(g_1, g_2, \ldots, g_{m_1}, g_{m_1+1}, \ldots, g_{m_1+m_2} \mid H_1) = \frac{1}{(\sqrt{2\pi}\sigma_1)^{m_1}} e^{-\frac{m_1}{2}} \cdot \frac{1}{(\sqrt{2\pi}\sigma_2)^{m_2}} e^{-\frac{m_2}{2}} \tag{3.8}$$

The likelihood ratio, L, is then defined as the ratio of the probability densities under the two hypotheses

$$L = \frac{p(g_1, g_2, \ldots \mid H_1)}{p(g_1, g_2, \ldots \mid H_0)} \tag{3.9}$$

$$= \frac{\sigma_0^{m_1+m_2}}{\sigma_1^{m_1} \cdot \sigma_2^{m_2}} \tag{3.10}$$

The values of the parameters σ_0, σ_1, and σ_2 in the above equation are estimated from Eqs. 3.2 and 3.3 using all the $(m_1 + m_2)$ pixels, m_1 pixels from region 1 and m_2 pixels from region 2, respectively. If the likelihood ratio L is below a threshold value, there is strong evidence for the likelihood that there is only one region and the two regions may be merged.

This approach may also be used for edge detection. Since the likelihood ratio indicates when two regions should be considered to be separate, it indicates when there should be a boundary between the two regions. For edge detection, the likelihood ratio between neighborhoods on either side of a point may be used to detect the presence of edges.

There are other possible modifications to this ratio which can play an important role in many applications. The likelihood ratio was derived under the assumption that a region contains constant gray values which, due to noise, have a normal distribution. It is possible to assume that the underlying intensity distribution is not constant but rather planar or quadratic. The likelihood ratio in these cases may be derived and used in a similar way.

3.4.2 Removing Weak Edges

Another approach to merging is to combine two regions if the boundary between them is weak. This approach attempts to remove weak edges between adjacent regions by considering not only the intensity characteristics, but also the length of the common boundary. The common boundary is dissolved if

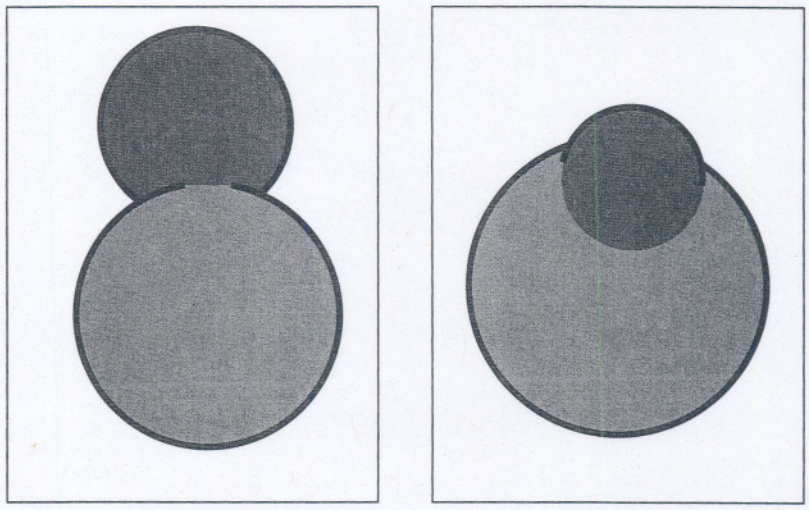


Figure 3.14: Approach 1: Merging by removing weak boundaries if the ratio of the weak boundary to the minimum region perimeter is above some threshold. *Left:* The two regions should *not* be merged because the weak boundary is very short as compared to the perimeter of the smaller region. *Right:* The two regions should be merged into a single region because the weak part of the common boundary is a significant fraction of the perimeter of the smaller region. Note that the strong boundary is marked by a bold line whereas the weak boundary is marked simply by a change in color of the two regions.

the boundary is weak and the resulting boundary (of the merged region) does not change gray value too quickly. The notation used in the following discussion is illustrated in Figures 3.14 and 3.15.

A weak boundary is one for which the intensities on either side differ by less than an amount T. Other criteria, such as edgeness values, may be used to determine the strength of an edge point that is on the boundary separating two regions. In an algorithm to merge regions by dissolving weak boundaries between regions, one must consider the relative lengths of the weak and complete boundaries between the regions. Two approaches which consider the relative lengths are:

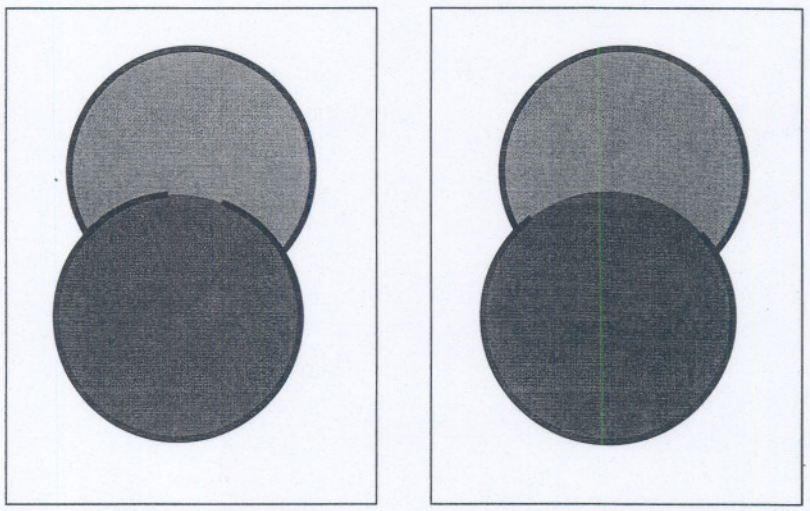


Figure 3.15: Approach 2: Merging by removing weak boundaries if the ratio of the weak boundary to the total common boundary is above some threshold. *Left:* The two regions should *not* be merged because the weak boundary is small when compared to the total common boundary. *Right:* The two regions should be merged into a single region because the weak part of the boundary is almost the same as the total common boundary. Note that the strong boundary is marked by a bold line whereas the weak boundary is marked simply by a change in color of the two regions.

1. Merge adjacent regions R_1 and R_2 if

$$\frac{W}{S} > \tau,$$

where W is the length of the weak part of the boundary, τ is a threshold, and $S = \min(S_1, S_2)$ is the minimum of the perimeters of the two regions as shown in Figure 3.14. In this algorithm, the performance depends on τ. For a small value of τ, there will be too many region merges. For a high value, the algorithm becomes too conservative. A good heuristic value for τ is 0.5.

2. Merge adjacent regions R_1 and R_2 if

$$\frac{W}{S} > \tau,$$

where S is now the common boundary as shown in Figure 3.15. The parameter value $\tau = 0.75$ usually yields satisfactory results.

3.4.3 Region Splitting

If some property of a region is not constant, the region should be split. The segmentation based on the split approach starts with large regions. In many cases, one may start with the whole image as the starting region. A split algorithm is given in Algorithm 3.6.

Several decisions must be made before a region is split. The problem is usually in deciding when a property is not constant over a region and how to split a region so that the property for each of the resulting components is constant. These questions are usually application-dependent and require knowledge of the characteristics of regions in that application. In some applications, the variance of the intensity values is used as a measure for how close the gray values are to being constant. In other applications, a function is fitted to approximate the underlying intensity values. The error between this function and the actual intensity values is used as the measure of region similarity.

More difficult than deciding if the gray values are constant across a region is deciding where to split a region. One approach used to determine the best boundary for dividing a region is to consider the measures of edge strength within the region. The easiest methods for splitting regions are those that divide the region into a fixed number of equal-sized regions; these are called regular decomposition methods. The quad tree representation for images, discussed in Section 3.3.2, is an example of regular decomposition.

Algorithm 3.6 Region Splitting

1. *Form initial regions in the image.*
2. *For each region in an image, recursively perform the following steps:*
   (a) *Compute the variance in the gray value for the region.*
   (b) *If the variance is above a threshold, split the region along the appropriate boundary.*

Algorithm 3.7 Split and Merge Region Segmentation

1. *Start with the entire image as a single region.*
2. *Pick a region R. If $P(R)$ is false, then split the region into four subregions.*
3. *Consider any two or more neighboring subregions, $R_1, R_2, \ldots, R_n$, in the image. If $P(R_1 \cup R_2 \cup \cdots \cup R_n)$ is true, merge the n regions into a single region.*
4. *Repeat these steps until no further splits or merges take place.*

The quad tree approach, however, must be modified for use in segmenting nonbinary (gray value) images. Instead of considering black and white regions, we must use the variance of the image intensity to decide whether a region should be split. Due to the numerous ways in which a region may be split, splitting regions is generally more difficult than merging them.

3.4.4 Split and Merge

Split and merge operations may be used together. After a presegmentation based on thresholding, a succession of splits and merges may be applied to refine the segmentation. Combined split and merge algorithms are useful for segmenting complex scenes. Domain knowledge may be introduced to guide the split and merge operations.

Suppose that an image is partitioned into a set of regions, $\{R_k\}$, for $k = 1, 2, \ldots, m$. All of the pixels in a region will be homogeneous according to some property defined by a predicate P applied to the region. The predicate represents the similarity between the pixels in a region. For example, the predicate could be defined using the variance in gray values within a region:

$$P(R) = \begin{cases} 1 & \text{if the variance is small} \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

The split and merge algorithm for region segmentation is outlined in Algorithm 3.7.

3.5 Region Growing

In many images, the gray values of individual regions are not nearly constant and more sophisticated techniques must be used for segmentation. The best techniques are those based on the assumption that the image can be partitioned into regions that can be modeled by simple functions. This idea can be applied naturally for region segmentation.

The segmentation problem set forth in Section 3.2 leads to an algorithm that starts with seed regions and then grows the regions to form larger regions satisfying these constraints. For the example in this text, the homogeneity predicate is based on fitting planar and biquadratic functions to the gray values in a region. However, in general, the homogeneity predicate can be based on any characteristic of the regions in the image such as average intensity, variance, texture, or color. The algorithm is given here as Algorithm 3.8.

The algorithm begins by partitioning the image into $n \times n$ regions where n is typically between 5 and 9. Regions are merged if a single planar or biquadratic function can be fit to the gray values in both regions. The planar and biquadratic models are a linear combination of basis functions. The basis functions span the variable-order bivariate polynomials, so the model is

$$f(x, y, a, m) = \sum_{i+j \leq m} a_{ij} x^i y^j, \tag{3.12}$$

where the order m of the model is restricted to $0 \leq m \leq 2$. This means that the region models are restricted to planar and biquadratic functions.

The homogeneity predicate is based on the distance of points in a region from the function that models the region:

$$\chi^2(R, a, m) = \sum_{(x,y) \in R} d^2(x, y, a, m) \tag{3.13}$$

where the distance is ordinary Euclidean distance:

$$d^2(x, y, a, m) = [g(x, y) - f(x, y, a, m)]^2 . \tag{3.14}$$

The gray value $g(x, y)$ at point (x, y) in the image plane is the gray value of the pixel at that image location. Given a set of points R, the problem is to find the order m of the model and the model parameters a that minimize the error function $\chi^2(R, a, m)$. This is a least-squares problem that can be

Algorithm 3.8 Region Growing Using Planar and Biquadratic Models

1. *Partition the image into initial seed regions* $R_i^{(0)}$.
2. *Fit a planar model to each seed region. If the chi-squared error is small enough, accept the seed region and its model; otherwise, reject the seed region.*
3. *For each region, find all points that are compatible with the region by extrapolating the region model to the neighbors of the region. Compatible points are defined as*

   $$C_i^{(k)} = \{(x,y) | d^2(x,y,a,m) \leq \epsilon \text{ and } (x,y) \text{ is a 4-neighbor of } R_i^{(k)} \cup C_i^{(k)}\}$$

   where ϵ *is the compatibility threshold.*
4. *If there were no compatible points, increase the order of the model:* $m \leftarrow m+1$. *If the model order is larger than the maximum model order, then do not grow the region further; otherwise, continue region growing by returning to step 3.*
5. *Form the new region* $R_i^{(k+1)} = R_i^{(k)} \cup C_i^{(k)}$, *refit the model at the same order to the new region, and compute the goodness of fit* $\chi^2(R_i^{(k+1)}, a, m)$.
6. *Compute the difference between the old and new goodness of fit for the region model:*

   $$\rho^{(k+1)} = \chi^2(m, a^{(k+1)}, R_i^{(k+1)}) - \chi^2(m, a^{(k)}, R_i^{(k)})$$

7. *If* $\rho^{(k+1)} < T_1$, *continue region growing by returning to step 3.*
8. *Increase the order of the model:* $m \leftarrow m+1$. *If the model order is larger than the maximum model order, then do not grow the region further.*
9. *Refit the region model at the new model order* m. *If the error of fit decreases, accept the new model order and continue region growing by returning to step 3; otherwise, do not grow region* R_i *further.*

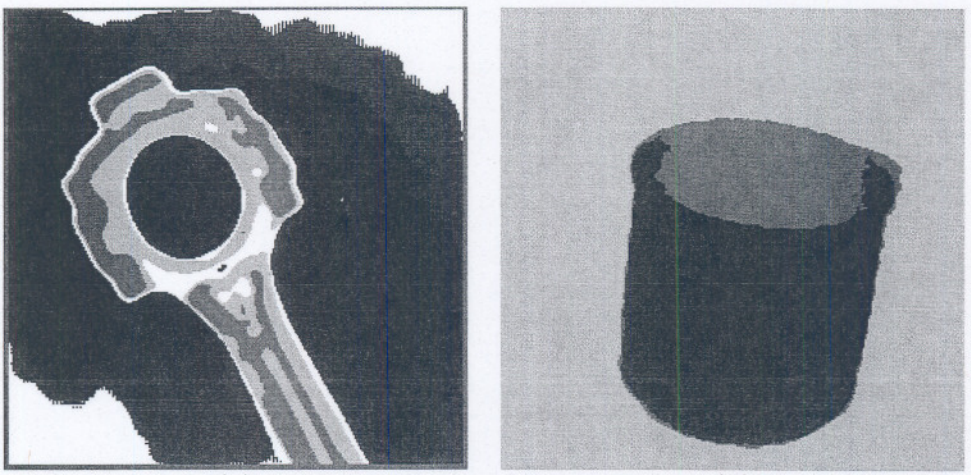
solved using singular value decomposition. Refer to [196] and Appendix B for a complete discussion on singular value decomposition.

The reason that the compatibility of points is checked against the surface patch before the surface is refit to the combined set of points is that least-squares fitting is very sensitive to outliers. If an outlier is added to the region before fitting, then the surface patch can be so severely distorted by the outlier that it no longer fits even the points that actually belong to the region.

It is not possible to know in advance how the image should be partitioned into regions that are modeled by distinct surface patches, since this is the segmentation problem itself. One can use a rigorous approach to find a conservative seed region. It is possible to use a conservative thresholding for finding the seed. One may also use a sophisticated approach to find initial seeds using the domain knowledge or the nature of images. For example, in range images differential geometric characteristics may be used to find initial seed regions. A computationally efficient approach to do this is to use some general image characteristics to identify such seed regions. One may partition the image into 7×7 seed regions and then fit surface patches to these seed regions. The seed regions are accepted based on the chi-squared error in the fit of the surface patch. If a 7×7 patch is rejected, then it is replaced by overlapping 5×5 surface patches to get higher resolution. Regions are grown by acquiring compatible points. Points are compatible with a region if the surface patch for the region can be extended to include the point such that the value of the point is not far from the surface. The surface patch is refit over the new domain which includes the original points plus the compatible points.

One may allow points to be associated with more than one region. The ambiguity is resolved by a postprocessing step that performs model selection. In classic region growing, each point is associated with at most one region. This constraint can be enforced by modifying the algorithm so that one region takes a point away from another region only if reassigning the point improves the fit of the surface patch model for both regions. More precisely, reassign a point only if the combined error for surface fits for both regions is lower. Figure 3.16 shows the results of this approach.

The classical region segmentation techniques described earlier can be viewed as region growing techniques where the surface patches are restricted to constants. In other words, the assumption is that regions are nearly constant



(a) (b)

Figure 3.16: The segmentation of an image using the region growing algorithm is given here. Part (a) shows the results for an intensity image and (b) shows results for a range image.

in image intensity and the image can be partitioned into regions of piecewise constant image intensity. More sophisticated segmentation techniques generalize the assumptions on the variation in image intensity to allow more realistic intensity variations due to shading.

Further Reading

After more than two decades of intensive research efforts, segmentation still remains a difficult problem in machine vision. Several approaches do not use any domain-specific information in the early stages of vision [163]. On the other hand, some psychophysicists believe that every aspect of perception uses domain information extensively [203]. Computer vision research has been influenced by both these views. Haralick and Shapiro [101] present a review of early segmentation techniques.

A discussion of thresholding methods for region formation is given in the books by Rosenfeld and Kak [206], and Haralick and Shapiro [103]. For an introduction to quad trees, see the book by Samet [210]. Pyramid representations are discussed in several sources; for an introduction see [230].

Region growing has been a popular topic. A statistical approach for

merging regions was proposed by Yakimovsky [258]. An algorithm for region growing was developed by Besl and Jain [28].

Color can be used for segmentation. A representative treatment of segmentation techniques using color information is given in [105]. For an early work based on histograms of colors, see [189]. Geman and Geman's [85] stochastic relaxation approach for segmentation and restoration of images has attracted significant attention. The relaxation method, also known as simulated annealing, has many applications in image analysis.

Explicit application of knowledge for segmentation has been addressed in many systems [30, 119, 165, 183]. Knowledge-based approaches definitely have strong appeal and will potentially be very useful for segmentation of complex images. So far their success has been limited due to poor performance of operators in domain-independent early processing. Model-based reasoning can help [49]. The model-based reasoning will emerge as a powerful approach in many applications of machine vision.

Considerable work on region growing for land use classification and related applications has been done in the field of remote sensing [208].

Exercises

3.1 Regions and boundaries may be considered duals of each other. They also contain equivalent information. Do you see advantages of one over the other in machine vision applications? Why?

3.2 In the automatic thresholding approach based on the mode method, the bottom of the valley is usually used as the threshold value. Assuming that the object and the background are of equal size in the image and their intensity values are drawn from two different Gaussian probability distributions $N(\mu_1, \sigma)$ and $N(\mu_2, \sigma)$, show that the recommended threshold indeed results in the optimal threshold because it minimizes the misclassification error. Obtain an expression for the optimal threshold for the general case when the object pixels cover $p\%$ of the image area and the object and background distributions are given by $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$, respectively.

3.3 What knowledge about the objects in images is used in the p-tile and mode methods for automatically selecting the threshold value?

3.4 What is the limitation of histogram-based methods for finding regions in an image? How can this be overcome in most applications? Can you suggest a postprocessing method for correcting the results obtained using a histogram-based method for images that contain objects that do not have Gaussian distributions for their intensity values?

3.5 Pyramid representations are appearing in many different forms. In many common multimedia applications thumbnail sketches are used. What are thumbnail sketches? Can you consider these schemes a subset of pyramid schemes?

3.6 Find the quad tree representation for the following object.

0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1

3.7 Kodak's photo-CD form of image representation has become popular in several applications. What is this representation? What are its useful features?

3.8 Quad tree representations have a serious limitation for applications in machine vision. A connected component in an image may be distributed at distant nodes in this representation. Can you suggest a scheme to modify quad trees to remove this limitation?

3.9 Another serious problem with quad trees is the rotation and scaling of regions. What will happen to the quad tree of an image if a region in it is translated, rotated, or scaled? Can you suggest some solutions to overcome these problems?

3.10 Show that the pyramid representation of an $n \times n$ image actually does fit within a linear array of size $2(2^{2\times\text{level}})$.

3.11 Assume that regions in an image have an intensity distribution that is planar combined with Gaussian noise. Thus, intensity value at a point of the region can be represented as

$$I(x, y) = Ax + By + C + N(0, \sigma^2). \tag{3.15}$$

Derive the likelihood ratio-based merge criterion under this assumption.

Computer Projects

3.1 Consider three different types of images: an office scene, an outdoor scene, and maybe a scene containing just one simple object. Implement an automatic thresholding scheme and study the results for these images.

3.2 Put two simple objects on a desk. Select one light and one dark object. Illuminate the scene such that light comes from the direction of the light object. Develop a suitable automatic thresholding scheme to segment the image as best as you can. Now change the direction of illumination by moving your light source so that the illumination is from the direction of the dark object. See whether your algorithm still works. If not, experiment and make it work in these and similar other cases.

3.3 Implement the likelihood ratio-based approach for merging regions. Use it as a postprocessing step in your earlier experiments.

3.4 Implement the region growing approach for finding regions in an image. Use it in your earlier experiments. Now compare the results of all these approaches and find the strengths and weaknesses of all these approaches for different situations in images.