16 Segmentation

16.1 Introduction

All image processing operations discussed in the preceding chapters aimed at a better recognition of objects of interest, i. e., at finding suitable local features that allow us to distinguish them from other objects and from the background. The next step is to check each individual pixel to see whether it belongs to an object of interest or not. This operation is called *segmentation* and produces a *binary image*. A pixel has the value one if it belongs to the object; otherwise it is zero. Segmentation is the operation at the threshold between *low-level image processing* and *image analysis*. After segmentation, we know which pixel belongs to which object. The image is parted into regions and we know the discontinuities as the boundaries between the regions. After segmentation, we can also analyze the shape of objects with operations such as those discussed in Chapter 19.

In this chapter, we discuss several types of elementary segmentation methods. Basically we can think of several basic concepts for segmentation. Pixel-based methods (Section 16.2) only use the gray values of the individual pixels. Region-based methods (Section 16.4) analyze the gray values in larger areas. Finally, edge-based methods (Section 16.3) detect edges and then try to follow them. The common limitation of all these approaches is that they are based only on local information. Even then they use this information only partly. Pixel-based techniques do not even consider the local neighborhood. Edge-based techniques look only for discontinuities, while region-based techniques analyze homogeneous regions. In situations where we know the geometric shape of an object, *model-based segmentation* can be applied (Section 16.5). We discuss an approach to the Hough transform that works directly from gray scale images (Section 16.5.3).

16.2 Pixel-Based Segmentation

Point-based or *pixel-based segmentation* is conceptually the simplest approach we can take for segmentation. We may argue that it is also the best approach. Why? The reason is that instead of trying a complex segmentation procedure, we should rather first use the whole palette

B. Jähne, Digital Image Processing
ISBN 3-540-24035-7

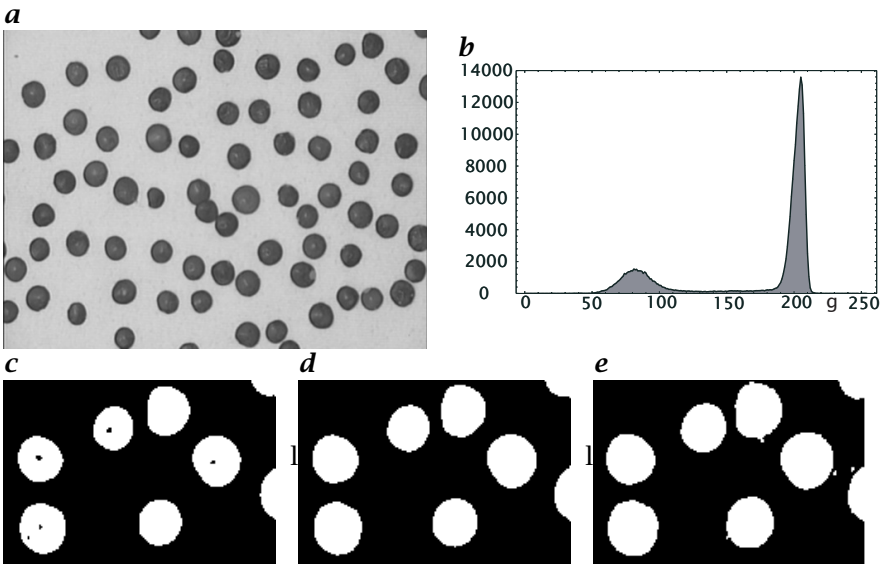

Figure 16.1: *Segmentation with a global threshold:* ***a*** *original image;* ***b*** *histogram;* ***c–e*** *upper right sector of* ***a*** *segmented with global thresholds of 110, 147, and 185, respectively.*

of techniques we have discussed so far in this book to extract those features that characterize an object in a unique way *before* we apply a segmentation procedure. It is always better to solve the problem at its root. If an image is unevenly illuminated, for instance, the first thing to do is to optimize the illumination of the scene. If this is not possible, the next step would be to identify the unevenness of the illumination system and to use corresponding image processing techniques to correct it. One possible technique has been discussed in Section 10.3.2.

If we have found a good feature to separate the object from the background, the histogram of this feature will show a *bimodal distribution* with two distinct maxima as in Fig. 16.1b. We cannot expect that the probability for gray values between the two peaks will be zero. Even if there is a sharp transition of gray values at the edge of the objects, there will always be some intermediate values due to a nonzero *point spread function* of the optical system and sensor (Sections 7.6.1 and 9.2.1). The smaller the objects are, the more area in the image is occupied by intermediate values filling the histograms in between the values for object and background (Fig. 16.1b).

How can we find an optimum threshold in this situation? In the case shown in Fig. 16.1, it appears to be easy because both the background and the object show rather uniform gray values. Thus we obtain a good segmentation for a large range of thresholds, between a low threshold of

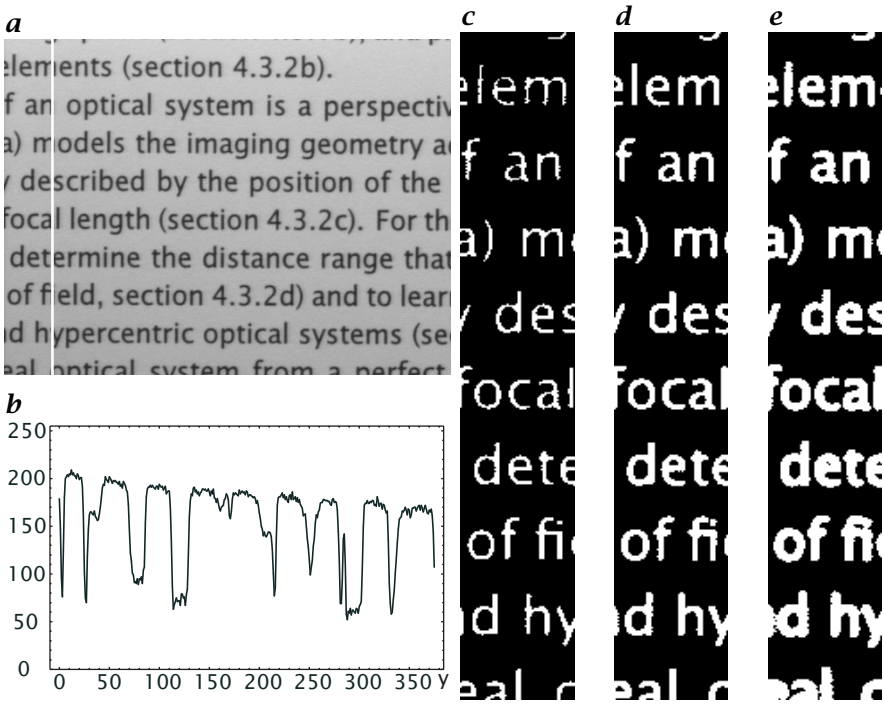

Figure 16.2: *Segmentation of an image with a graded background:* ***a*** *original image;* ***b*** *profile of column 55 (as marked in* ***a****);* ***c–e*** *first 64 columns of* ***a*** *segmented with global thresholds of 90, 120, and 150, respectively.*

110, where the objects start to get holes (Fig. 16.1c), and a high threshold of 185, close to the value of the background, where some background pixels are detected as object pixels.

However, a close examination of Fig. 16.1c–e reveals that the size of the segmented objects changes significantly with the level of the threshold. Thus it is critical for a bias-free determination of the geometrical features of an object to select the correct threshold. This cannot be performed without knowledge about the type of the edge between the object and the background. In the simple case of a symmetrical edge, the correct threshold is given by the mean gray value between the background and the object pixels.

This strategy fails as soon as the background is not uniform, or if objects with different gray values are contained in the image (Figs. 16.2 and 16.3). In Fig. 16.2b, the segmented letters are thinner in the upper, brighter part of the image. Such a bias is acceptable for some applications such as the recognition of typeset letters. However, it is a serious flaw for any gauging of object sizes and related parameters.

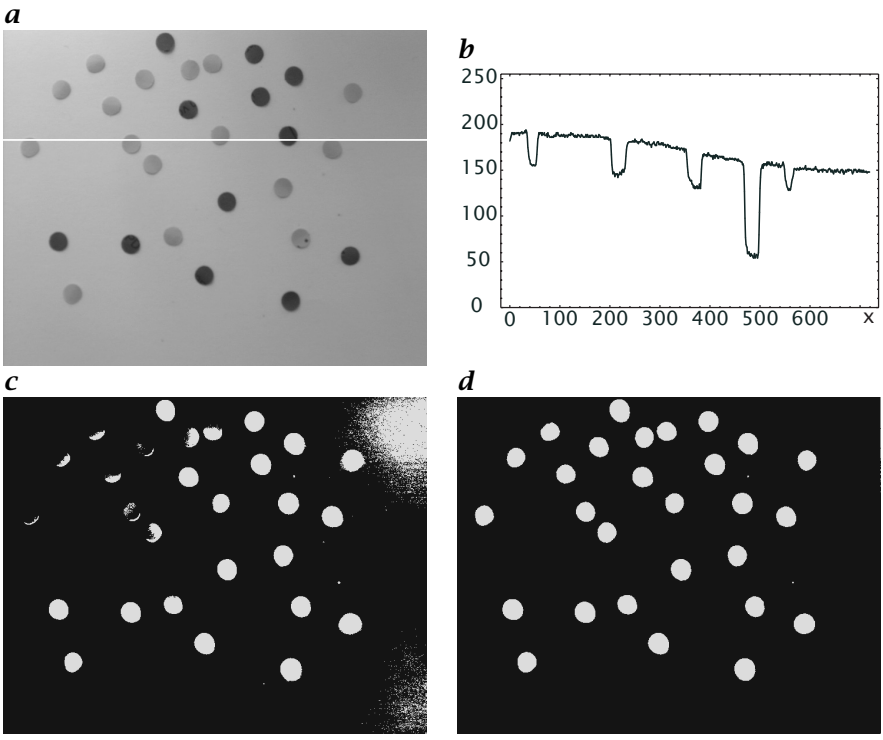

Figure 16.3: *Segmentation of an image with an uneven illumination:* ***a*** *original image with inhomogeneous background illumination (for histogram, see Fig. 10.10b);* ***b*** *profile of row 186 (as marked in* ***a****);* ***c*** *and* ***d*** *segmentation results with an optimal global threshold of the images in* ***a*** *before and after the image is first corrected for the inhomogeneous background (Fig. 10.10c), respectively.*

Figure 16.3a shows an image with two types of circles; both are circles but of different brightness. The radiance of the brighter circles comes close to the background. Indeed, a histogram (Fig. 10.10b) shows that the gray values of these brighter circles no longer form a distinct maximum but overlap with the wide distribution of the background.

Consequently, the global thresholding fails (Fig. 16.3c). Even with an optimal threshold, some of the background in the right upper and lower corners are segmented as objects and the brighter circles are still segmented only partly. If we first correct for the inhomogeneous illumination as illustrated in Fig. 10.10, all objects are segmented perfectly (Fig. 16.3d). We still have the problem, however, that the areas of the dark circles are too large because the segmentation threshold is too close to the background intensity.

16.3 Edge-Based Segmentation

16.3.1 Principle

We have seen in Section 16.2 that even with perfect illumination, pixel-based segmentation results in a bias of the size of segmented objects when the objects show variations in their gray values (Figs. 16.2 and 16.3). Darker objects will become too small, brighter objects too large. The size variations result from the fact that the gray values at the edge of an object change only gradually from the background to the object value. No bias in the size occurs if we take the mean of the object and the background gray values as the threshold. However, this approach is only possible if all objects show the same gray value or if we apply different thresholds for each objects.

An *edge-based segmentation* approach can be used to avoid a bias in the size of the segmented object without using a complex thresholding scheme. Edge-based segmentation is based on the fact that the position of an edge is given by an extreme of the first-order derivative or a zero crossing in the second-order derivative (Fig. 12.1). Thus all we have to do is to search for local maxima in the edge strength and to trace the maximum along the edge of the object.

16.3.2 Bias by Uneven Illumination

In this section, we study the bias of various segmentation techniques by a nonuniform background and varying object brightness. We assume that the object edge can be modeled adequately by a step edge that is blurred by a point spread function $h(x)$ with even symmetry. For the sake of simplicity, we model the 1-D case. Then the brightness of the object in the image with an edge at the origin can be written as

$$g(x) = g_0 \int_{-\infty}^{x} h(x)\mathrm{d}x \quad \text{with} \quad \int_{-\infty}^{\infty} h(x)\mathrm{d}x = 1. \tag{16.1}$$

We further assume that the background intensity can be modeled by a parabolic variation of the form

$$b(x) = b_0 + b_1 x + b_2 x^2. \tag{16.2}$$

Then the total intensity in the image is given by

$$g(x) = g_0 \int_{-\infty}^{x} h(x)\mathrm{d}x + b_0 + b_1 x + b_2 x^2. \tag{16.3}$$

The first and second derivatives are

$$\begin{aligned} g_x(x) &= g_0 h(x) + b_1 + 2b_2 x, \\ g_{xx}(x) &= g_0 h_x(x) + 2b_2. \end{aligned} \tag{16.4}$$

Around the maximum, we can approximate the point spread function $h(x)$ by a parabola: $h(x) \approx h_0 - h_2 x^2$. Then

$$\begin{aligned} g_x(x) &\approx g_0 h_0 - g_0 h_2 x^2 + b_1 + 2 b_2 x, \\ g_{xx}(x) &\approx -2 g_0 h_2 x + 2 b_2. \end{aligned} \tag{16.5}$$

The position of the edge is given as the zero of the second derivative. Therefore the bias in the edge position estimation, x_b, is given from Eq. (16.5) as

$$x_b \approx \frac{b_2}{g_0 h_2}. \tag{16.6}$$

From Eq. (16.6) we can conclude:

1. Edge-based segmentation shows no bias in the edge position even if the background intensity is sloped.
2. Edge-based segmentation shows no bias with the intensity g_0 of the edge as it is the case with intensity-based segmentation (Section 16.2).
3. Edge-based segmentation is only biased by a curvature in background intensity. The bias is directly related to the ratio of the curvature in the background intensity to the maximum curvature of the point spread function. This means that the bias is higher for blurred edges. The bias is also inversely proportional to the intensity of the object and thus seriously affects only objects with weak contrast.

16.3.3 Edge Tracking

Edge-based segmentation is a sequential method. In contrast to pixel-based and most region-based segmentations, it cannot be performed in parallel on all pixels. The next step to be performed rather depends on the results of the previous steps. A typical approach is as described in the following. The image is scanned line by line for maxima in the magnitude of the gradient. When a maximum is encountered, a *tracing algorithm* tries to follow the maximum of the gradient around the object until it reaches the starting point again. Then a search begins for the next maximum in the gradient. Like region-based segmentation, edge-based segmentation takes into account that an object is characterized by adjacent pixels.

16.4 Region-Based Segmentation

16.4.1 Principles

Region-based methods focus our attention on an important aspect of the segmentation process we missed with point-based techniques. There we classified a pixel as an object pixel judging solely on its gray value

independently of the context. This meant that isolated points or small areas could be classified as object pixels, disregarding the fact that an important characteristic of an object is its *connectivity*.

In this section we will not discuss such standard techniques as split-and-merge or region-growing techniques. Interested readers are referred to Rosenfeld and Kak [172] or Jain [97]. Here we discuss rather a technique that aims to solve one of the central problems of the segmentation process.

If we use not the original image but a feature image for the segmentation process, the features represent not a single pixel but a small neighborhood, depending on the mask sizes of the operators used. At the edges of the objects, however, where the mask includes pixels from both the object and the background, any feature that could be useful cannot be computed. The correct procedure would be to limit the mask size at the edge to points of either the object or the background. But how can this be achieved if we can only distinguish the object and the background after computation of the feature?

Obviously, this problem cannot be solved in one step, but only iteratively using a procedure in which feature computation and segmentation are performed alternately. In principle, we proceed as follows. In the first step, we compute the features disregarding any object boundaries. Then we perform a preliminary segmentation and compute the features again, now using the segmentation results to limit the masks of the neighborhood operations at the object edges to either the object or the background pixels, depending on the location of the center pixel. To improve the results, we can repeat feature computation and segmentation until the procedure converges into a stable result.

16.4.2 Pyramid Linking

Burt [18] suggested a *pyramid-linking* algorithm as an effective implementation of a combined segmentation feature computation algorithm. We will demonstrate it using the illustrative example of a noisy step edge (Fig. 16.4). In this case, the computed feature is simply the mean gray value. The algorithm includes the following steps:

1. *Computation of the Gaussian pyramid.* As shown in Fig. 16.4a, the gray values of four neighboring pixels are averaged to form a pixel on the next higher level of the pyramid. This corresponds to a smoothing operation with a box filter.
2. *Segmentation by pyramid-linking.* As each pixel contributes to either of two pixels on the higher level, we can now decide to which it most likely belongs. The decision is simply made by comparing the gray values and choosing the pixel next to it. The link is pictured in Fig. 16.4b by an edge connecting the two pixels. This procedure is

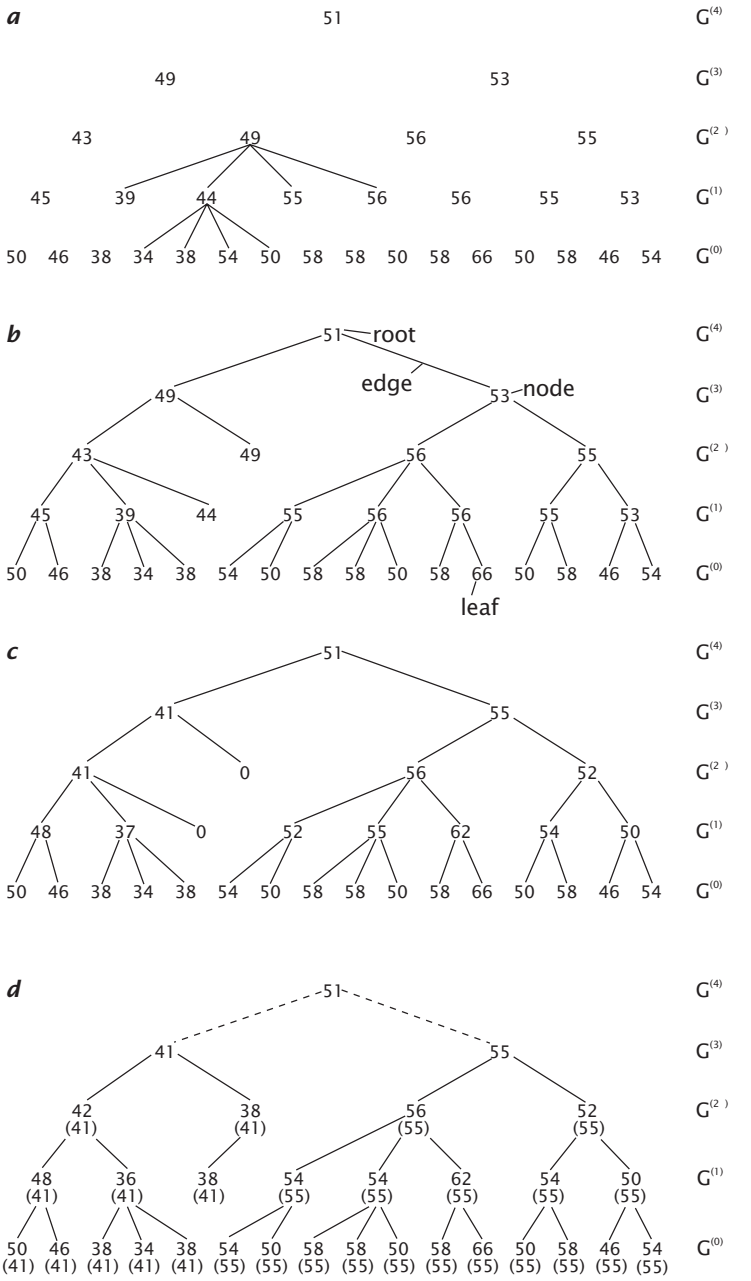

Figure 16.4: *Pyramid-linking segmentation procedure with a one-dimensional noisy edge:* ***a*** *computation of the Gaussian pyramid;* ***b*** *node-linking;* ***c*** *re-computation of the mean gray values;* ***d*** *final result after several iterations of steps* ***b*** *and* ***c*** *.*

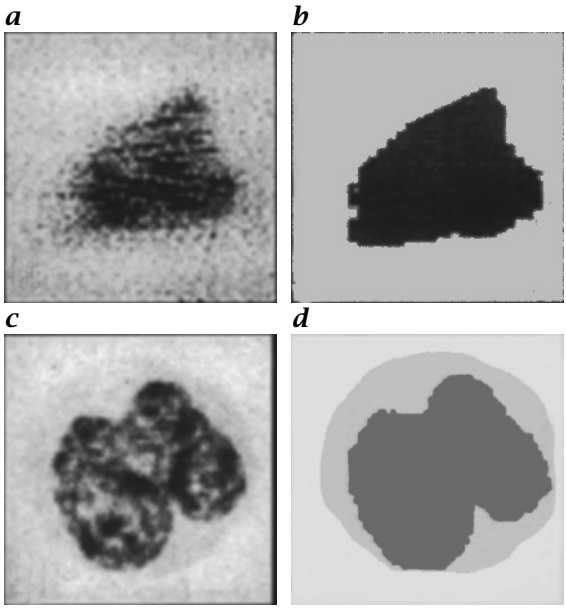

Figure 16.5: *Noisy* ***a*** *tank and* ***c*** *blood cell images segmented with the pyramid-linking algorithm in* ***b*** *two and* ***d*** *three regions, respectively; after Burt [18].*

repeated through all the levels of the pyramid. As a result, the links in the pyramid constitute a new data structure. Starting from the top of the pyramid, one pixel is connected with several pixels on the next lower level. Such a data structure is called a *tree* in computer science. The links are called *edges*; the data points are the gray values of the pixels and are denoted as *nodes* or *vertices*. The node at the highest level is called the *root* of the tree and the nodes with no further links are called the *leaves* of the tree. A node linked to a node at a lower level is denoted as the *father node* of this node. Correspondingly, each node linked to a node at a higher level is defined as the *son node* of this node.

3. *Averaging of linked pixels.* Next, the resulting link structure is used to recompute the mean gray values, now using only the linked pixels (Fig. 16.4c), i. e., the new gray value of each father node is computed as the average gray value of all the son nodes. This procedure starts at the lowest level and is continued through all the levels of the pyramid.

The last two steps are repeated iteratively until we reach a stable result shown in Fig. 16.4d. An analysis of the link-tree shows the result of the segmentation procedure. In Fig. 16.4d we recognize two *subtrees*, which have their roots in the third level of the pyramid. At the next lower level, four subtrees originate. But the differences in the gray values at

this level are significantly smaller. Thus we conclude that the gray value structure is obviously parted into two regions. Then we obtain the final result of the segmentation procedure by transferring the gray values at the roots of the two subtrees to the linked nodes at the lowest level. These values are shown as braced numbers in Fig. 16.4d.

The application of the pyramid-linking segmentation algorithm to two-dimensional images is shown in Fig. 16.5. Both examples illustrate that even very noisy images can be successfully segmented with this procedure. There is no restriction on the form of the segmented area.

The pyramid-linking procedure merges the segmentation and the computation of mean features for the objects extracted in an efficient way by building a tree on a pyramid. It is also advantageous that we do not need to know the number of segmentation levels beforehand. They are contained in the structure of the tree. Further details of pyramid-linking segmentation are discussed in Burt et al. [20] and Pietikäinen and Rosenfeld [153].

16.5 Model-Based Segmentation

16.5.1 Introduction

All segmentation techniques discussed so far utilize only local information. In Section 1.6 (Fig. 1.16) we noted the remarkable ability of the human vision system to recognize objects even if they are not completely represented. It is obvious that the information that can be gathered from local neighborhood operators is not sufficient to perform this task. Instead we require specific knowledge about the geometrical shape of the objects, which can then be compared with the local information.

This train of thought leads to *model-based segmentation*. It can be applied if we know the exact shape of the objects contained in the image. We consider here only the simplest case: straight lines.

16.5.2 Parameter Space; Hough Transform

The approach discussed here detects lines even if they are disrupted by noise or are only partially visible. We start by assuming that we have a segmented image that contains lines of this type. The fact that points lie on a straight line results in a powerful constraint that can be used to determine the parameters of the straight line. For all points $[x_n, y_n]^T$ on a straight line, the following condition must be met:

$$y_n = a_0 + a_1 x_n, \tag{16.7}$$

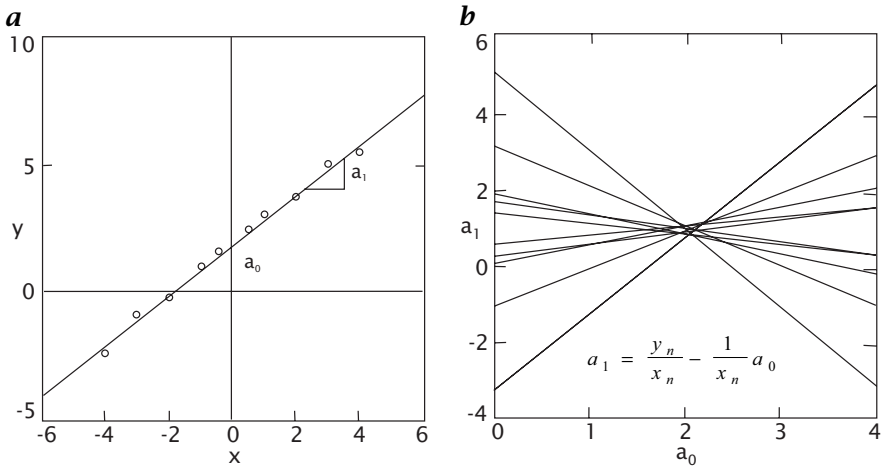

Figure 16.6: *Hough transform for straight lines: the $[x, y]^T$ data space (**a**) is mapped onto the $[a_0, a_1]^T$ model space (**b**).*

where a_0 and a_1 are the offset and slope of the line. We can read Eq. (16.7) also as a condition for the parameters a_0 and a_1:

$$a_1 = \frac{y_n}{x_n} - \frac{1}{x_n} a_0. \tag{16.8}$$

This is again the equation for a line in a new space spanned by the parameters a_0 and a_1. In this space, the line has the offset y_n/x_n and a slope of $-1/x_n$.

With one point given, we already cease to have a free choice of a_0 and a_1 as the parameters must satisfy Eq. (16.8).

The space spanned by the model parameters a_0 and a_1 is called the *model space*. Each point reduces the model space to a line. Thus, we can draw a line in the model space for each point in the data space, as illustrated in Fig. 16.6. If all points lie on a straight line in the data space, all lines in the model space meet in one point which gives the parameters a_0 and a_1 of the lines. As a line segment contains many points, we obtain a reliable estimate of the two parameters of the line. In this way, a line in the data space is mapped onto a point in the model space. This transformation from the data space to the model space via a model equation is called the *Hough transform*. It is a versatile instrument to detect lines even if they are disrupted or incomplete.

In practical applications, the well-known equation of a straight line given by Eq. (16.7) is not used. The reason is simply that the slope of a line may become infinite and is thus not suitable for a discrete model space. A better parameterization of a straight line is given by using two different parameters with finite values. One possibility is to take the

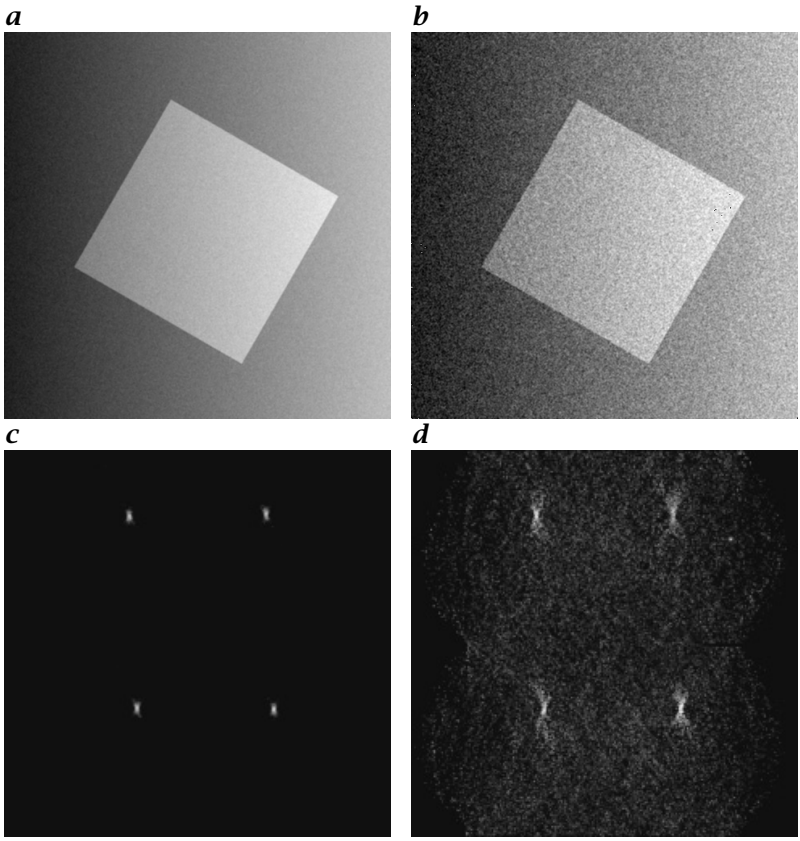

Figure 16.7: *Orientation-based fast Hough transform:* ***a*** *and* ***b*** *unevenly illuminated noisy squares;* ***c*** *and* ***d*** *Hough model space with the distance d (horizontal axis) and the angle θ (vertical axis) of the lines according to Eq. (16.9) for* ***a*** *and* ***b****, respectively.*

angle of the slope of the line and the distance of the line from the center of the coordinate system. With these two parameters, the equation of a straight line can be written as

$$\tilde{\boldsymbol{n}}\boldsymbol{x} = d \quad \text{or} \quad x\cos\theta + y\sin\theta = d, \tag{16.9}$$

where $\tilde{\boldsymbol{n}}$ is a vector normal to the line and θ the angle of this vector to the x axis of the image coordinate system.

The drawback of the Hough transform method for line detection is the high computational effort. For each point in the image, we must compute a line in the parameter space and increment each point in the model space through which the line passes.

16.5.3 Orientation-Based Fast Hough Transform

A significant speed-up of the Hough transform can be obtained by using additional information from low-level image processing. The analysis of local neighborhoods with the *structure tensor* method not only detects edges but also gives their slope. Therefore, we have two pieces of information for each point in the image if it lies on an edge: a point through which the edge passes and its orientation. This already completely describes the line.

Consequently, each point on a line in the image space corresponds no longer to a line — as discussed in Section 16.5.2 — but to a single point in the parameter space. The one-to-one correspondence considerably speeds up the computation of the Hough transform. For each point in the image, we only need to add *one* point to the parameter space.

An application of the orientation-based Hough transform is demonstrated in Fig. 16.7. Figure 16.7a, b shows noisy images with a square. To extract the edges of the rectangle, no segmentation is required. We just compute the components of the structure tensor with the techniques described in Section 13.3.6. Then for each point in the image, θ and d are computed according to Eq. (16.9).

As a weighting factor for the contribution of a point to the parameter space, we use the length of the orientation vector. In this way, points are weighted according to the certainty measure for the local orientation and thus the edge strength.

In the Hough parameter space (Fig. 16.7c and d), four clusters show up, corresponding to the four different lines of the square. The clusters occur in pairs as two lines are parallel to each other and differ only by the distance to the center of the image. Note how well the technique works even at high noise levels.

16.6 *Exercises*

16.1: Simple segmentation methods

Interactive demonstration of simple segmentation methods (dip6ex16.01).

16.2: Hough transform

Interactive demonstration of the Hough transform (dip6ex16.02)

16.3: **Segmentation with constant background

All segmentation methods are faced with the problem of systematic errors. Assume that an image contains objects with different, but constant brightness. The background has a constant brightness h. For the following computations it is sufficient to use two objects with brightnesses g_1 and g_2. The objects have a width $l > 5$ and are convolved by a rectangular point spread function with 5 pixel width during the image acquisition

process. The image signal contains an additive zero mean white noise with a variance σ^2.
Three segmentation approaches are available:

P Pixel-based segmentation with a constant global threshold at the brightness level t,

G Edge-based segmentation on the base of first-order derivative filters. The edge position is given by the maximum value of the magnitude of the gradient.

L Edge-based segmentation on the base of second-order derivative filters. The edge position is given by zero crossings of the Laplacian operator.

Answer the following questions for the three segmentation methods:

1. Which brightness difference is required in order to distinguish the objects from the background in a statistically significant way? The difference between thresholds and signal levels should be at least three times the standard deviation σ of the noise.
2. Is it possible that one of the methods causes a systematic error in the size of the object? If yes, compute the systematic error and compare it for the different methods.

16.4: *Segmentation with varying background

Answer the same questions as for Exercise 16.3 with the following image model: An object with a constant brightness g and an inhomogeneous background with a quadratic change:

$$h = h_0 + h_1 x + h_2 x^2$$

(Hint: It is sufficient to discuss the problem in one dimension.)

16.7 *Further Readings*

Pitas [154, Chapter 6] and Umbaugh [203, Section 2.4] deal with various standard algorithms for segmentation. Forsyth and Ponce [54, Kapitel 14] discusses segmentation by Clustering.