3

Binary Images: Geometric Properties

In this chapter we explore black-and-white (two-valued) images. These are easier to acquire, store, and process than images in which many brightness levels are represented. However, since binary images encode only information about the silhouette of an object, they are useful only in restricted situations. We discuss the conditions that must be satisfied for the successful application of binary image-processing techniques. In this chapter we concentrate on simple geometric properties such as image area, position, and orientation. These quantities are useful in directing the interaction of a mechanical manipulator with a part, for example. Other aspects of binary images, such as methods for iterative modification, are explored in the next chapter.

Since images contain a great deal of information, issues of representation become important. We show that the geometric properties of interest can be computed from projections of binary images. These projections are a great deal easier to store and to process. In much of this chapter we deal with continuous binary images, where a characteristic function is either zero or one at every point in the image plane. This makes the presentation simpler; but when we use a digital computer, the image must be divided into picture cells. The chapter concludes with a discussion of discrete binary images and ways of exploiting their spatial coherence to

reduce transmission and storage costs, as well as processing time.

3.1 Binary Images

Let us start by considering a single object in the field of view, with everything else taken as "background." If the object appears consistently darker (or brighter) than the background, it is easy to define a *characteristic function* $b(x, y)$ that is zero for all image points corresponding to the background and one for points on the object (figure 3-1). Such a two-valued function, called a *binary image*, can be obtained by *thresholding* the gray-level image. The thresholding operation simply defines the characteristic function to be zero where the brightness is greater than some threshold value and one where it is not (or vice versa).

Sometimes it is convenient to think of the image components, as well as the holes in them, as sets. This will allow us to combine images using set operations such as union and intersection. At other times it is convenient to use point-by-point Boolean operations, such as logical *and* ($\wedge$) and logical *or* ($\vee$). These are just two different ways of looking at the same basic operations.

For a given image size, a binary image is easier to digitize, store, and transmit than a full gray-level image, since it contains about an order of magnitude fewer bits. Naturally, some information is lost and fewer options are open to us in processing such an image. In fact, we can present a fairly complete theory of what can and cannot be done with binary images, something that still eludes us in the case of gray-level images.

First of all, we can compute various geometric properties, such as the size and position of the object in the image. If there is more than one object in the field of view, we can determine some topological properties of the assembly, such as the difference between the number of objects and the number of holes. It is also possible to label the individual objects and to perform the geometrical computations for each one separately. Finally, we can simplify binary images for further processing by modifying them in an iterative fashion.

Binary image processing is well understood and lends itself to high-speed hardware implementation, but one must keep in mind its limitations. We have already mentioned the need for high object-to-background contrast. Moreover, the pattern obtained must be essentially two-dimensional. Remember that all we have is the outline or silhouette of the object. This gives little information about the object's overall shape or its attitude in space.

The characteristic function $b(x, y)$ has a value for each point in the image. We call this a *continuous* binary image. Later, we shall consider discrete binary images, obtained by tessellating the image plane suitably.

3.2 Simple Geometrical Properties

Assume for now that we have a single object in the field of view. The area of the object, given the characteristic function $b(x, y)$, can be found as follows:

$$A = \iint_I b(x, y)\, dx\, dy,$$

where the integration is over the whole image I. If there is more than one object, this formula will give the total area.

3.2.1 Area and Position

How do we determine the position of the object in the image? Since the object is usually not just a single point, we must give a precise meaning to the term "position." The usual practice is to choose the center of area as the representative point (figure 3-2). The center of area is the center of

mass of a figure of the same shape with constant mass per unit area. The center of mass, in turn, is that point where all the mass of the object could be concentrated without changing the first moment of the object about any axis. In the two-dimensional case the moment about the x-axis is

$$\overline{x} \iint_I b(x,y)\,dx\,dy = \iint_I x\,b(x,y)\,dx\,dy,$$

and the moment about the y-axis is

$$\overline{y} \iint_I b(x,y)\,dx\,dy = \iint_I y\,b(x,y)\,dx\,dy,$$

where $(\overline{x},\overline{y})$ is the position of the center of area. The integrals appearing on the left of these equations are, of course, just the area A, as noted above. To find $\overline{x}$ and $\overline{y}$ we must assume that A is not equal to zero. Note, by the way, that A is the zeroth moment of $b(x,y)$.

3.2.2 Orientation

We also want to determine how the object lies in the field of view, that is, its *orientation*. This is a little harder. Let us assume that the object

is somewhat elongated; then the orientation of the axis of elongation can be used to define the orientation of the object (figure 3-3). (There is a remaining two-way ambiguity, however, as discussed in exercise 3-7.) How precisely do we define the direction in which the object is elongated? The usual practice is to choose the axis of least second moment. This is the two-dimensional equivalent of the axis of least inertia. We find the line for which the integral of the square of the distance to points in the object is a minimum; that integral is

$$E = \iint_I r^2 b(x, y)\, dx\, dy,$$

where r is the perpendicular distance from the point (x, y) to the line sought

after.

To choose a particular line in the plane, we need to specify two parameters. A convenient pair consists of the distance ρ from the origin to the closest point on the line, and the angle θ between the x-axis and the line, measured counterclockwise (figure 3-4). We prefer these parameters because they change continuously when the coordinate system is translated or rotated. There are no problems when the line is parallel, or nearly parallel, to either of the coordinate axes.

Using these parameters, we can write the equation of the line in the form

$$x \sin\theta - y \cos\theta + \rho = 0$$

by noting that the line intersects the x-axis at $-\rho/\sin\theta$ and the y-axis at $+\rho/\cos\theta$. The closest point on the line to the origin is at $(-\rho\sin\theta, +\rho\cos\theta)$. We can write parametric equations for points on the line as follows:

$$x_0 = -\rho\sin\theta + s\cos\theta \qquad \text{and} \qquad y_0 = +\rho\cos\theta + s\sin\theta,$$

where s is the distance along the line from the point closest to the origin.

Given a point (x, y) on the object, we need to find the closest point (x_0, y_0) on the line, so that we can compute the distance r between the point and the line (figure 3-5). Clearly

$$r^2 = (x - x_0)^2 + (y - y_0)^2.$$

Substituting the parametric equations for x_0 and y_0 we obtain

$$r^2 = (x^2 + y^2) + \rho^2 + 2\rho(x \sin\theta - y \cos\theta) - 2s(x \cos\theta + y \sin\theta) + s^2.$$

Differentiating with respect to s and setting the result equal to zero lead to

$$s = x \cos\theta + y \sin\theta.$$

This result can now be substituted back into the parametric equations for x_0 and y_0. From these we can compute the differences

$$\begin{aligned} x - x_0 &= +\sin\theta \, (x \sin\theta - y \cos\theta + \rho), \\ y - y_0 &= -\cos\theta \, (x \sin\theta - y \cos\theta + \rho), \end{aligned}$$

and thus

$$r^2 = (x \sin\theta - y \cos\theta + \rho)^2.$$

Comparing this result with the equation for the line, we see that the line is the locus of points for which $r = 0$! Conversely, it becomes apparent that the way we chose to parameterize the line has the advantage of giving us

the distance from the line directly.

Finally, we can now address the minimization of

$$E = \iint_I (x \sin\theta - y \cos\theta + \rho)^2 \, b(x,y) \, dx \, dy.$$

Differentiating with respect to ρ and setting the result to zero lead to

$$A(\overline{x} \sin\theta - \overline{y} \cos\theta + \rho) = 0,$$

where $(\overline{x}, \overline{y})$ is the center of area defined above. Thus the axis of least second moment passes through the center of area. This suggests a change of coordinates to $x' = x - \overline{x}$ and $y' = y - \overline{y}$, for

$$x \sin\theta - y \cos\theta + \rho = x' \sin\theta - y' \cos\theta,$$

and so

$$E = a \sin^2\theta - b \sin\theta \cos\theta + c \cos^2\theta,$$

where a, b, and c are the second moments given by

$$a = \iint_{I'} (x')^2 \, b(x,y) \, dx' \, dy',$$

$$b = 2 \iint_{I'} (x'y') \, b(x,y) \, dx' \, dy',$$

$$c = \iint_{I'} (y')^2 \, b(x,y) \, dx' \, dy'.$$

Now we can write the formula for E in the form

$$E = \frac{1}{2}(a+c) - \frac{1}{2}(a-c)\cos 2\theta - \frac{1}{2} b \sin 2\theta.$$

Differentiating with respect to θ and setting the result to zero, we have

$$\tan 2\theta = \frac{b}{a-c},$$

unless $b = 0$ and $a = c$. Consequently

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2 + (a-c)^2}} \quad \text{and} \quad \cos 2\theta = \pm \frac{a-c}{\sqrt{b^2 + (a-c)^2}}.$$

Of the two solutions, the one with the plus signs in the expressions for $\sin 2\theta$ and $\cos 2\theta$ leads to the desired minimum for E. Conversely, the solution with minus signs in the two expressions corresponds to the maximum value for E. (This can be shown by considering the second derivative of E with respect to θ.) If $b = 0$ and $a = c$, we find that E is independent of θ. In

this case, the object is too symmetric to allow us to define an axis in this way. The ratio of the smallest value for E to the largest tells us something about how "rounded" the object is. This ratio is zero for a straight line and one for a circle.

Another way to look at the problem is to seek the rotation θ that makes the 2×2 matrix of second moments diagonal. We show in exercise 3-5 that the calculation above essentially amounts to finding the eigenvalues and eigenvectors of the 2×2 matrix of second moments given by

$$\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}.$$

3.3 Projections

The position and orientation of an object can be calculated using first and second moments only. (There is a remaining two-way ambiguity, however, as discussed in exercise 3-7.) We do not need the original image to obtain the first and second moments; projections of the image are sufficient. This is of interest since the projections are more compact and suggest much

faster algorithms.

Consider a line through the origin inclined at an angle θ with respect to the x-axis (figure 3-6). Now construct a new line, intersecting the original line at right angles, at a point that is a distance t from the origin. Let distance along the new line be designated s. The integral of $b(x, y)$ along the new line gives one value of the *projection*. That is,

$$p_\theta(t) = \int_L b(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\, ds.$$

The integration is carried out over the portion of the line L that lies within the image. The vertical projection ($\theta = 0$), for example, is simply

$$v(x) = \int_L b(x, y)\, dy,$$

while the horizontal projection ($\theta = \pi/2$) is

$$h(y) = \int_L b(x, y)\, dx.$$

Now, since

$$A = \iint_I b(x, y)\, dx\, dy,$$

we have

$$A = \int v(x)\, dx = \int h(y)\, dy.$$

More important,

$$\overline{x}A = \iint_I x\, b(x, y)\, dx\, dy = \int x\, v(x)\, dx$$

and

$$\overline{y}A = \iint_I y\, b(x, y)\, dx\, dy = \int y\, h(y)\, dy.$$

Thus the first moments of the projections are equal to the first moments of the original image.

To compute orientation we need the second moments, too. Two of these are easy to compute from the projections:

$$\iint_I x^2\, dx\, dy = \int x^2 v(x)\, dx \qquad \text{and} \qquad \iint_I y^2\, dx\, dy = \int y^2 h(y)\, dy.$$

We cannot, however, compute the integral of the product xy from the two projections introduced so far. We can add the diagonal projection

$(\theta = \pi/4)$,

$$d(t) = \int b\left(\frac{1}{\sqrt{2}}(t-s), \frac{1}{\sqrt{2}}(t+s)\right)\, ds.$$

Now consider that

$$\iint_I \frac{1}{2}(x+y)^2\, b(x,y)\, dx\, dy = \iint_I t^2\, b(x,y)\, ds\, dt = \int t^2\, d(t)\, dt,$$

and that

$$\iint_I \frac{1}{2}(x+y)^2\, b(x,y)\, dx\, dy = \iint_I \left(\frac{1}{2}x^2 + xy + \frac{1}{2}y^2\right)\, b(x,y)\, dx\, dy.$$

So

$$\iint_I xy\, b(x,y)\, dx\, dy = \int t^2 d(t)\, dt - \frac{1}{2}\int x^2 v(x)\, dx - \frac{1}{2}\int y^2 h(y)\, dy.$$

Thus all the integrals needed for the computation of the position and orientation of a region in the binary image can be found from the horizontal,

diagonal, and vertical projections (figure 3-7).

It is interesting to compare tomographic reconstruction with what we are doing here. In *tomography* one obtains integrals of the absorbing density of the material in some body. The problem is to reconstruct the distribution of the material. In the case of X-ray tomography, the body is commonly treated as a stack of slices, and each slice is considered a separate two-dimensional problem. The projections obtained are analogous to the projections discussed above. It can be shown that the distribution of absorbing material can be determined if we have the projections for all directions, provided that the density is zero outside some bounded region.

How does tomographic reconstruction relate to what we are doing here? The main difference is that in our case the function over which the line integrals are taken can only have values zero and one. The other difference is that we are not capturing sufficient information to reconstruct the original binary image, but only enough to determine the center of area and axis of least inertia. There are some interesting unsolved problems in the reconstruction of binary images from projections. Suppose, for example, that there is only one region that is nonzero and that this region is convex. How many projections are required to determine fully the outline of the region? Further, are there any noncircular shapes that have the same horizontal

and vertical projections as a circle?

3.4 Discrete Binary Images

So far we have dealt with continuous binary images, defined at all points in the image plane. It should be obvious that the integrals become sums when we turn to discrete binary images (figure 3-8). The area, for example, can be computed (in units of the area of a picture cell) using the sum

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij},$$

where b_{ij} is the value of the binary image at the point in the i^{th} row and the j^{th} column. Here we have assumed that the image has been digitized on a square grid with n rows and m columns.

Usually the image is scanned out row by row, in the same sequence as the beam on a television set explores the screen (except for interlace). As each picture cell is read, we check whether $b_{ij} = 1$. If so, we add 1, i, j, i^2, ij, and j^2 to the accumulated totals for the area, first moments, and second moments, as appropriate. At the end of the scan, the area, position, and orientation can be easily calculated from these totals.

3.5 Run-Length Coding

For binary images there are several coding techniques that can compress the data even further. A popular one is *run-length coding*. This method exploits the fact that along any particular scan line there will usually be long *runs* of zeros or ones. Instead of transmitting the individual bits, then, we can send numbers indicating the lengths of such runs. The run-length code for the image line

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

is just $[1, 4, 3, 2, 4]$. Some special code may be used to indicate the beginning of each line. Further, a convention is followed as to whether the line starts with a zero or a one. If the line starts the other way, an initial run-length code of zero is sent.

Let us use the convention that r_{ik} is the k^{th} run of the i^{th} line and that the first run in each row is a run of zeros. (Thus all the even runs will correspond to ones in the image.) Suppose further that there are m_i runs on the i^{th} line. (This number will be odd if the last run contains zeros.) How do we use run-length codes to rapidly compute the desired

geometric properties? Clearly the area is just the sum of the run lengths corresponding to ones in the image:

$$A = \sum_{i=1}^{n} \sum_{k=1}^{m_i/2} r_{i,2k}.$$

Note that the sum is over the even runs only. The center of area is a little harder to calculate. First of all, we obtain the horizontal projection (figure 3-9) as follows:

$$h_i = \sum_{k=1}^{m_i/2} r_{i,2k}.$$

From this we can easily compute the vertical position $\bar{i}$ of the center of area using

$$A\bar{i} = \sum_{i=1}^{n} i\, h_i.$$

But what about the vertical projection v_j? Remember that a particular entry in this projection is obtained by adding up all picture cell values in one column of the image. It is therefore hard to compute the vertical

projection directly from the run lengths.

Now consider instead the first difference of the vertical projection,

$$\overline{v}_j = v_j - v_{j-1}, \quad \text{with } \overline{v}_1 = v_1.$$

Clearly it can be obtained by projecting not the image data, but the first horizontal differences of the image data:

$$\overline{b}_{i,j} = b_{i,j} - b_{i,j-1}.$$

The first difference $\overline{b}_{ij}$ has the advantage over b_{ij} itself that it is nonzero only at the beginning of each run. It equals $+1$ where the data change from a 0 to a 1, and -1 where the data change from a 1 to a 0 (figure 3-10). We can locate these places by computing the summed run-length code $\widetilde{r}_{ik}$:

$$\widetilde{r}_{ik} = 1 + \sum_{l=1}^{k} r_{il}$$

or, recursively, $\widetilde{r}_{i,0} = 1$ and $\widetilde{r}_{i,j+1} = \widetilde{r}_{i,j} + r_{i,j}$. Then, for a transition at $j = \widetilde{r}_{ik}$ we add $(-1)^{k+1}$ to the accumulated total for the first difference of the vertical projection $\overline{v}_j$.

From this first difference we can compute the vertical projection itself using the simple summation

$$v_j = \sum_{l=1}^{j} \overline{v}_l,$$

or, recursively, $v_0 = 0$ and $v_{j+1} = v_j + \overline{v}_{j+1}$. The computation is illustrated figure 3-10. Given the vertical projection, we can easily compute the horizontal position $\overline{j}$ of the center of area using

$$A\overline{j} = \sum_{j=1}^{m} j\, v_j.$$

We shall show in exercise 3-8 that the diagonal projection can be obtained in a way similar to that used to obtain the vertical projection.

3.6 References

Another discussion of binary image processing can be found in *Digital Picture Processing* by Rosenfeld & Kak [1982]. The book edited by Stoffel, *Graphical and Binary Image Processing and Applications* [1982], contains some interesting recent work in binary image processing. To see implementations of some of the algorithms discussed here, look in the chapter on

binary image processing in *Lisp*, by Winston & Horn [1984]. Many interesting binary images produced by artists can be found in *Silhouettes—A Pictorial Archive of Varied Illustrations*, edited by Grafton [1979].

Run-length coding only exploits redundancy along one dimension. There have been several attempts to make use of spatial coherence in two dimensions in order to reduce transmission and storage costs. Perhaps the most successful scheme of this nature so far is that developed by IBM and described by Mitchell & Goertzel [1979]. Further references relating to binary image processing may be found at the end of the next chapter.

Simon Kahan of Bell Labs has solved one of the problems at the end of section 3.3. He has proven that no noncircular shape has the same horizontal and vertical projections as a circle.

3.7 Exercises

3-1 The extreme values of the moment of inertia can be written as

$$E = \frac{1}{2}(a+c) \pm \frac{1}{2}\sqrt{b^2 + (a-c)^2},$$

where a, b, and c are as defined in this chapter. Show that $E \geq 0$. When is $E = 0$?

3-2 Rewrite the expression for the second moment about an axis inclined by an angle θ,

$$E = \frac{1}{2}(a+c) - \frac{1}{2}(a-c)\cos 2\theta - \frac{1}{2}b\sin 2\theta,$$

in the form

$$E = \frac{1}{2}(a+c) + \sqrt{b^2 - (a-c)^2}\cos 2(\theta - \phi).$$

What is the angle ϕ?

3-3 Let a, b, and c be the integrals defined in this chapter.

(a) Write a, b, and c in terms of integrals over x and y, instead of x' and y'.

(b) Find expressions for $\sin\theta$ and $\cos\theta$ from the expressions given for $\sin 2\theta$ and $\cos 2\theta$.

3-4 It is useful, at times, to replace a shape with one that is simpler but has the same zeroth, first, and second moments. Consider a region whose boundary is an ellipse with semimajor axis α along the x-axis and semiminor axis β along the y-axis. The equation of this ellipse is

$$\left(\frac{x}{\alpha}\right)^2 + \left(\frac{y}{\beta}\right)^2 = 1.$$

Show that the minimum and maximum second moments of the region about an axis through the origin are

$$\frac{\pi}{4}\alpha\beta^3 \quad \text{and} \quad \frac{\pi}{4}\beta\alpha^3,$$

respectively. Now the second moment of any region about an axis inclined at an angle θ can be written in the form

$$E = a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta.$$

Compute the major and minor axes of an equivalent ellipse, that is, one that has the same second moment about any axis through the origin.

3-5 A vector $\mathbf{v}$ with the property that $\mathbf{Mv} = \lambda\mathbf{v}$ for some λ is called an *eigenvector* of the matrix $\mathbf{M}$. The constant multiplier λ is the corresponding *eigenvalue*.

(a) Show that the symmetric 2×2 matrix

$$\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$$

has two real eigenvalues given by

$$\frac{1}{2}(a+c) \pm \frac{1}{2}\sqrt{b^2 + (a-c)^2}.$$

Hint: A homogeneous set of linear equations has a solution only when the determinant of the coefficient matrix equals zero.

(b) Show that the corresponding eigenvectors are given by

$$\left(\sqrt{\sqrt{(a-c)^2+b^2} - (c-a)}, +\sqrt{\sqrt{(a-c)^2+b^2} - (a-c)}\right)^T,$$

and

$$\left(\sqrt{\sqrt{(a-c)^2+b^2} - (a-c)}, -\sqrt{\sqrt{(a-c)^2+b^2} - (c-a)}\right)^T.$$

(c) Show that the two eigenvectors are orthogonal and have magnitude

$$\sqrt{2\sqrt{(a-c)^2+b^2}}.$$

(d) Relate these results to the problem of finding the axis of least inertia.

3-6 Show how the integral

$$\iint_I xy\, b(x,y)\, dx\, dy$$

can be computed from the horizontal and vertical projections and a projection $p_\theta(t)$ at angle θ, where

$$p_\theta(t) = \int b(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\, ds.$$

What restriction must be placed on θ ?

3-7 The axis of least inertia uniquely defines a line through an object. The methods introduced so far do not, however, assign an orientation to this line. This two-way ambiguity in determining orientation may be significant in some applications. One can use auxiliary measurements to provide the needed extra bit of information. For example, it may be helpful to know where the point on the silhouette is that is furthest from the center of area.

Alternatively, we can consider computing higher-order moments. How many projections are required to find all moments up to the third? Which projections are most conveniently added to the ones we already have, that is, the horizontal ($\rightarrow$), the vertical ($\downarrow$), and the diagonal from NW to SE ($\searrow$)?

3-8 Show exactly how diagonal projections in the NW-to-SE ($\searrow$) and NE-to-SW ($\swarrow$) directions can be computed from the run lengths.

3-9 If the scan line is very long, many bits must be reserved for each run-length number (for $4,096$ picture cells, 12-bit numbers are needed). This is wasteful if most runs are short. Devise a scheme in which shorter numbers are sent.

3-10 Can the run-length code ever be less compact than the original image? If so, suggest preprocessing methods that might alleviate the problem.

3-11 Assuming a fixed picture, estimate how the number of bits increases when the number of rows and the number of columns are doubled. Consider both straightforward binary encoding as well as run-length coding of the image.

3-12 Suppose we have the following runs, r_{ik}:

1	4	2	5	2
1	8	1	3	1
2	6	2	2	2

Find the summed run-length codes $\overline{r}_{ik}$. From these find $\overline{v}_j$, the first difference of the vertical projection. Show that the vertical projection v_j is

0	2	3	3	3	2	2	3	2	1	3	3	1	0

3-13 In this chapter we have assumed that a binary image is represented by a matrix of zeros and ones. We might equally well use a list of the boundary curves of the regions in the image. Each boundary could be approximated by a polygon, for example. Show that the area of a region R can be computed as follows:

$$A = + \oint_{\partial R} x \, dy = - \oint_{\partial R} y \, dx,$$

where the integrals are taken in a counterclockwise fashion around the boundary ∂R of the region R. Develop similar integrals for the first and second moments needed in the computation of the center of area and the axis of least inertia. Hint: Use Gauss's integral theorem for converting area integrals into loop integrals.