CHAPTER

High-level feature extraction: fixed shape matching

5

CHAPTER OUTLINE HEAD

Feature Extraction & Image Processing for Computer Vision.

5.1 Overview

High-level *feature extraction* concerns finding shapes and objects in computer images. To be able to recognize human faces automatically, for example, one approach is to extract the component features. This requires extraction of, say, the eyes, the ears, and the nose, which are the major face features. To find them, we can use their shape: the white part of the eyes is ellipsoidal; the mouth can appear as two lines, as do the eyebrows. Alternatively, we can view them as objects and use the low-level features to define collections of points which define the eyes, nose, and mouth, or even the whole face. This feature extraction process can be viewed as similar to the way we perceive the world: many books for babies describe basic geometric shapes such as triangles, circles, and squares. More complex pictures can be decomposed into a structure of simple shapes. In many applications, analysis can be guided by the way the shapes are arranged. For the example of face image analysis, we expect to find the eyes above (and either side of) the nose and we expect to find the mouth below the nose.

In feature extraction, we generally seek **invariance properties** so that the extraction result does not vary according to chosen (or specified) conditions. This implies finding objects, whatever their position, their orientation, or their size. That is, techniques should find shapes reliably and robustly whatever the value of any parameter that can control the appearance of a shape. As a basic **invariant**, we seek immunity to changes in the **illumination** level: we seek to find a shape whether it is light or dark. In principle, as long as there is contrast between a shape and its background, the shape can be said to exist and can then be detected. (Clearly, any computer vision technique will fail in extreme lighting conditions; you cannot see anything when it is completely dark.) Following illumination, the next most important parameter is **position**: we seek to find a shape wherever it appears. This is usually called *position*, *location*, or *translation invariance*. Then, we often seek to find a shape irrespective of its **rotation** (assuming that the object or the camera has an unknown orientation): this is usually called *rotation* or *orientation invariance*. Then, we might seek to determine the object at whatever **size** it appears, which might be due to physical change, or to how close the object has been placed to the camera. This requires *size* or *scale invariance*. These are the main invariance properties we shall seek from our shape extraction techniques. However, nature (as usual) tends to roll balls under our feet: there is always **noise** in images. Also since we are concerned with shapes, note that there might be more than one in the image. If one is on top of the other, it will **occlude**, or hide, the other, so not all the shape of one object will be visible.

But before we can develop image analysis techniques, we need techniques to extract the shapes and objects. Extraction is more complex than **detection**, since extraction implies that we have a description of a shape, such as its position and size, whereas detection of a shape merely implies knowledge of its existence within an image. This chapter concerns shapes which are fixed in shape (such as

Table 5.1 Overview of Chapter 5

Main Topic	Subtopics	Main Points
Pixel operations	How we detect features at a **pixel** level. What are the **limitations** and **advantages** of this approach. Need for **shape** information.	*Thresholding*. *Differencing*.
Template matching	Shape extraction by **matching**. Advantages and disadvantages. Need for **efficient** implementation.	*Template matching*. Direct and Fourier implementations. Noise and occlusion.
Low-level features	Collecting **low-level features** for object extraction. **Frequency-**based and **parts-**based approaches. Detecting **distributions** of measures.	*Wavelets* and *Haar* wavelets. *SIFT* and *SURF* descriptions and *Histogram of oriented gradients*.
Hough transform	Feature extraction by **matching**. Hough transforms for **conic sections**. Hough transform for **arbitrary shapes**. **Invariant** formulations. Advantages in **speed** and **efficacy**.	Feature extraction by evidence gathering. *Hough transforms* for lines, circles, and ellipses. *Generalized* and *invariant Hough transforms*.

a segment of bone in a medical image); the following chapter concerns shapes which can deform (like the shape of a walking person).

The techniques presented in this chapter are outlined in Table 5.1. We first consider whether we can detect objects by thresholding. This is only likely to provide a solution when illumination and lighting can be controlled, so we then consider two main approaches: one is to extract constituent parts and the other is to extract constituent shapes. We can actually collect and describe low-level features described earlier. In this, wavelets can provide object descriptions, as can scale-invariant feature transform (SIFT) and distributions of low-level features. In this way we represent objects as a collection of interest points, rather than using shape analysis. Conversely, we can investigate the use of shape: t**emplate matching** is a model-based approach in which the shape is extracted by searching for the best correlation between a known model and the pixels in an image. There are alternative ways to compute the correlation between the template and the image. Correlation can be implemented by considering the image or frequency domains and the template can be defined by considering intensity values or a binary shape. The **Hough transform** defines an efficient implementation of template matching for binary templates. This technique is capable of extracting simple shapes such as lines and quadratic forms as well as arbitrary shapes. In any case, the

complexity of the implementation can be reduced by considering invariant features of the shapes.

5.2 Thresholding and subtraction

Thresholding is a simple shape extraction technique, as illustrated in Section 3.3.4, where the images could be viewed as the result of trying to separate the eye from the background. If it can be assumed that the shape to be extracted is defined by its brightness, then thresholding an image at that brightness level should find the shape. Thresholding is clearly sensitive to change in illumination: if the image illumination changes so will the perceived brightness of the target shape. Unless the threshold level can be arranged to adapt to the change in brightness level, any thresholding technique will fail. Its attraction is **simplicity**: thresholding does not require much computational effort. If the illumination level changes in a linear fashion, using histogram equalization will result in an image that does not vary. Unfortunately, the result of histogram equalization is sensitive to noise, shadows, and variant illumination: noise can affect the resulting image quite dramatically and this will again render a thresholding technique useless. Let us illustrate this by considering Figure 5.1 and let us consider trying to find either the ball or the player, or both in Figure 5.1(a). Superficially, these are the brightest objects so one value of the threshold (Figure 5.1(b)) finds the player's top, shorts and socks, and the ball—but it also finds the text in the advertising and the goalmouth. When we increase the threshold (Figure 5.1(c)), we lose parts of the player but still find the goalmouth. Clearly we need to include more knowledge or to process the image more.

Thresholding after **intensity normalization** (Section 3.3.2) is less sensitive to noise, since the noise is stretched with the original image and cannot affect the stretching process much. However, it is still sensitive to shadows and variant illumination. Again, it can only find application where the illumination can be carefully controlled. This requirement is germane to any application that uses basic

(a) Image

(b) Low threshold

(c) High threshold

FIGURE 5.1

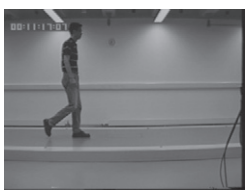
Extraction by thresholding.

thresholding. If the overall illumination level cannot be controlled, it is possible to threshold edge magnitude data since this is insensitive to overall brightness level, by virtue of the implicit differencing process. However, edge data is rarely continuous and there can be gaps in the detected perimeter of a shape. Another major difficulty, which applies to thresholding the brightness data as well, is that there are often more shapes than one. If the shapes are on top of each other, one occludes the other and the shapes need to be separated.

An alternative approach is to **subtract** an image from a known background before thresholding. This assumes that the background is known precisely, otherwise many more details than just the target feature will appear in the resulting image; clearly the subtraction will be unfeasible if there is **noise** on either image and especially on both. In this approach, there is no implicit shape description, but if the thresholding process is sufficient, it is simple to estimate basic shape parameters, such as position.

The subtraction approach is illustrated in Figure 5.2. Here, we seek to separate or extract a walking subject from their background. When we subtract the background of Figure 5.2(b) from the image itself, we obtain most of the subject with some extra background just behind the subject's head (this is due to the effect of the moving subject on **lighting**). Also, removing the background removes some of the subject: the horizontal bars in the background have been removed from the subject by the subtraction process. These aspects are highlighted in the thresholded image (Figure 5.2(c)). It is not particularly a poor way of separating the subject from the background (we have the subject but we have chopped through his midriff), but it is not especially good either. So it does provide an estimate of the object, but an estimate is only likely to be reliable when the lighting is highly controlled. (A more detailed separation of moving objects from their static background, including estimation of the background itself, is found in Chapter 9.)

Even though thresholding and subtraction are attractive (because of simplicity and hence their speed), the performance of both techniques is sensitive to partial shape data, to noise, to variation in illumination, and to occlusion of the target



(a) Image of walking subject

(b) Background

(c) After background subtraction and thresholding

FIGURE 5.2

Shape extraction by subtraction and thresholding.

shape by other objects. Accordingly, many approaches to image interpretation use higher level information in shape extraction, namely how the pixels are connected. This can resolve these factors.

5.3 Template matching

5.3.1 Definition

Template matching is conceptually a simple process. We need to match a **template** to an image, where the template is a sub-image that contains the shape we are trying to find. Accordingly, we center the template on an image point and count up how many points in the template **matched** those in the image. The procedure is repeated for the entire image and the point which led to the best match, the maximum count, is deemed to be the point where the shape (given by the template) lies within the image.

Consider that we want to find the template of Figure 5.3(b) in the image of Figure 5.3(a). The template is first positioned at the origin and then matched with the image to give a count which reflects how well the template matched that part of the image at that position. The count of matching pixels is increased by one for each point where the brightness of the template matches the brightness of the image. This is similar to the process of template convolution, as illustrated in Figure 3.11. The difference here is that points in the image are matched with those in the template, and the sum is of the number of matching points as opposed to the weighted sum of image data. The best match is when the template is placed at the position where the rectangle is matched to itself. Obviously, this process

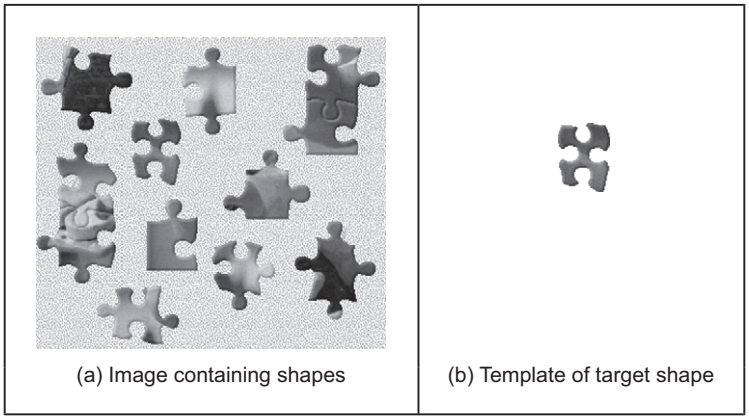


FIGURE 5.3

Illustrating template matching.

can be generalized to find, for example, templates of different **size** or **orientation**. In these cases, we have to try all the templates (at expected rotation and size) to determine the best match.

Formally, template matching can be defined as a method of parameter estimation. The parameters define the position (and pose) of the template. We can define a template as a discrete function $\mathbf{T}_{x,y}$. This function takes values in a window. That is, the coordinates of the points $(x,y) \in \mathbf{W}$. For example, for a 2×2 template, we have the set of points $\mathbf{W} = \{(0,0),(0,1),(1,0),(1,1)\}$.

Let us consider that each pixel in the image $\mathbf{I}_{x,y}$ is corrupted by additive Gaussian noise. The noise has a mean value of zero and the (unknown) standard deviation is σ. Thus, the probability that a point in the template placed at coordinates (i,j) matches the corresponding pixel at position $(x,y) \in \mathbf{W}$ is given by the normal distribution

$$p_{i,j}(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y}}{\sigma}\right)^2} \tag{5.1}$$

Since the noise affecting each pixel is independent, the probability that the template is at position (i,j) is the combined probability of each pixel that the template covers. That is,

$$L_{i,j} = \prod_{(x,y)\in\mathbf{W}} p_{i,j}(x,y) \tag{5.2}$$

By substitution of Eq. (5.1), we have

$$L_{i,j} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\frac{1}{2}\sum_{(x,y)\in\mathbf{W}}\left(\frac{\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y}}{\sigma}\right)^2} \tag{5.3}$$

where n is the number of pixels in the template. This function is called the **likelihood** function. Generally, it is expressed in logarithmic form to simplify the analysis. Note that the logarithm scales the function, but it does not change the position of the maximum. Thus, by taking the logarithm, the likelihood function is redefined as

$$\ln(L_{i,j}) = n \ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2}\sum_{(x,y)\in\mathbf{W}}\left(\frac{\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y}}{\sigma}\right)^2 \tag{5.4}$$

In *maximum likelihood estimation*, we have to choose the parameter that maximizes the likelihood function, i.e., the positions that minimize the rate of change of the objective function:

$$\frac{\partial \ln(L_{i,j})}{\partial i} = 0 \quad \text{and} \quad \frac{\partial \ln(L_{i,j})}{\partial j} = 0 \tag{5.5}$$

That is,

$$\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y}) \frac{\partial \mathbf{I}_{x+i,y+j}}{\partial i} = 0$$
$$\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y}) \frac{\partial \mathbf{I}_{x+i,y+j}}{\partial j} = 0 \tag{5.6}$$

We can observe that these equations are also the solution of the minimization problem given by

$$\min e = \sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \mathbf{T}_{x,y})^2 \tag{5.7}$$

That is, maximum likelihood estimation is equivalent to choosing the template position that minimizes the squared error (the squared values of the differences between the template points and the corresponding image points). The position where the template best matches the image is the estimated position of the template within the image. Thus, if you measure the match using the squared error criterion, then you will be choosing the *maximum likelihood* solution. This implies that the result achieved by template matching is optimal for images corrupted by Gaussian noise. A more detailed examination of the method of least squares is given in Appendix 2, Section 11.2. (Note that the *central limit theorem* suggests that practically experienced noise can be assumed to be Gaussian distributed though many images appear to contradict this assumption.) Of course you can use other error criteria such as the absolute difference rather than the squared difference or, if you feel more adventurous, you might consider robust measures such as M-estimators.

We can derive alternative forms of the squared error criterion by considering that Eq. (5.7) can be written as

$$\min e = \sum_{(x,y)\in \mathbf{W}} \mathbf{I}^2_{x+i,y+j} - 2\mathbf{I}_{x+i,y+j}\mathbf{T}_{x,y} + \mathbf{T}^2_{x,y} \tag{5.8}$$

The last term does not depend on the template position (i,j). As such, it is constant and cannot be minimized. Thus, the optimum in this equation can be obtained by minimizing

$$\min e = \sum_{(x,y)\in \mathbf{W}} \mathbf{I}^2_{x+i,y+j} - 2 \sum_{(x,y)\in \mathbf{W}} \mathbf{I}_{x+i,y+j}\mathbf{T}_{x,y} \tag{5.9}$$

If the first term

$$\sum_{(x,y)\in \mathbf{W}} \mathbf{I}^2_{x+i,y+j} \tag{5.10}$$

is approximately constant, then the remaining term gives a measure of the similarity between the image and the template. That is, we can maximize the

cross-correlation between the template and the image. Thus, the best position can be computed by

$$\max e = \sum_{(x,y)\in \mathbf{W}} \mathbf{I}_{x+i,y+j}\mathbf{T}_{x,y} \tag{5.11}$$

However, the squared term in Eq. (5.10) can vary with position, so the match defined by Eq. (5.11) can be poor. Additionally, the range of the cross-correlation is dependent on the size of the template and it is noninvariant to changes in image lighting conditions. Thus, in an implementation, it is more convenient to use either Eq. (5.7) or (5.9) (in spite of being computationally more demanding than the cross-correlation in Eq. (5.11)). Alternatively, cross-correlation can be **normalized** as follows. We can rewrite Eq. (5.8) as

$$\min e = 1 - 2\frac{\sum_{(x,y)\in \mathbf{W}} \mathbf{I}_{x+i,y+j}\mathbf{T}_{x,y}}{\sum_{(x,y)\in \mathbf{W}} \mathbf{I}^2_{x+i,y+j}} \tag{5.12}$$

Here the first term is constant and thus the optimum value can be obtained by

$$\max e = \frac{\sum_{(x,y)\in \mathbf{W}} \mathbf{I}_{x+i,y+j}\mathbf{T}_{x,y}}{\sum_{(x,y)\in \mathbf{W}} \mathbf{I}^2_{x+i,y+j}} \tag{5.13}$$

In general, it is convenient to normalize the gray level of each image window under the template. That is,

$$\max e = \frac{\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \overline{\mathbf{I}}_{i,j})(\mathbf{T}_{x,y} - \overline{\mathbf{T}})}{\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \overline{\mathbf{I}}_{i,j})^2} \tag{5.14}$$

where $\overline{\mathbf{I}}_{i,}j$ is the mean of the pixels $\mathbf{I}_{x+i,y+j}$ for points within the window (i.e., $(x,y)\in\mathbf{W}$) and $\overline{\mathbf{T}}$ is the mean of the pixels of the template. An alternative form to Eq. (5.14) is given by **normalizing** the cross-correlation. This does not change the position of the optimum and gives an interpretation as the normalization of the cross-correlation vector. That is, the cross-correlation is divided by its modulus. Thus,

$$\max e = \frac{\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \overline{\mathbf{I}}_{i,j})(\mathbf{T}_{x,y} - \overline{\mathbf{T}})}{\sqrt{\sum_{(x,y)\in \mathbf{W}} (\mathbf{I}_{x+i,y+j} - \overline{\mathbf{I}}_{i,j})^2 (\mathbf{T}_{x,y} - \overline{\mathbf{T}})^2}} \tag{5.15}$$

However, this equation has a similar computational complexity to the original formulation in Eq. (5.7).

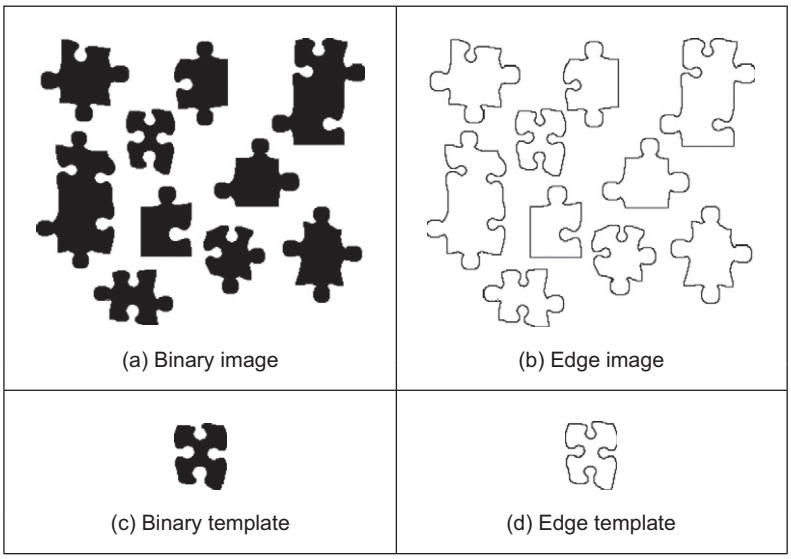


FIGURE 5.4

Example of binary and edge template matching.

A particular implementation of template matching is when the image and the template are binary. In this case, the binary image can represent regions in the image or it can contain the edges. These two cases are illustrated in the example shown in Figure 5.4. The advantage of using binary images is that the amount of **computation** can be **reduced**. That is, each term in Eq. (5.7) will take only two values: it will be one when $\mathbf{I}_{x+i,y+j} = \mathbf{T}_{x,y}$ and zero otherwise. Thus, Eq. (5.7) can be implemented as

$$\max e = \sum_{(x,y)\in \mathbf{W}} \overline{\mathbf{I}_{x+i,y+j} \oplus \mathbf{T}_{x,y}} \tag{5.16}$$

where the symbol $\overline{\oplus}$ denotes the exclusive NOR operator. This equation can be easily implemented and requires significantly less resource than the original matching function.

Template matching develops an *accumulator space* that stores the match of the template to the image at different locations; this corresponds to an implementation of Eq. (5.7). It is called an accumulator, since the match is **accumulated** during application. Essentially, the accumulator is a 2D array that holds the difference between the template and the image at different positions. The position in the image gives the same position of match in the accumulator. Alternatively, Eq. (5.11) suggests that the peaks in the accumulator resulting from template correlation give the location of the template in an image: the coordinates of the point of best match. Accordingly, template correlation and template matching can be viewed as similar processes. The location of a template can be determined by

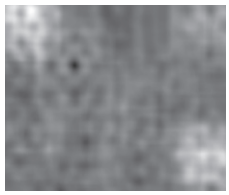
either process. The binary implementation of template matching (Eq. (5.16)) is usually concerned with thresholded edge data. This equation will be reconsidered in the definition of the Hough transform, the topic of the following section.

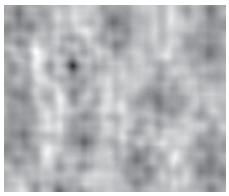
The Matlab code to implement template matching is the function `TMatching` given in Code 5.1. This function first clears an accumulator array, `accum`, then searches the whole picture, using pointers `i` and `j`, and then searches the whole template for matches, using pointers `x` and `y`. Note that the position of the template is given by its center. The accumulator elements are incremented according to Eq. (5.7). The accumulator array is delivered as the result. The match for each position is stored in the array. After computing all the matches, the minimum element in the array defines the position where most pixels in the template matched those in the image. As such, the minimum is deemed to be the coordinates of the point where the template's shape is most likely to lie within the original image. It is possible to implement a version of template matching without the accumulator array, by storing the location of the minimum alone. This will give the same result though it requires little storage. However, this implementation will provide

```
%Template Matching Implementation

function accum=TMatching(inputimage,template)

%Image size & template size
  [rows,columns]=size(inputimage);
  [rowsT,columnsT]=size(template);

  %Centre of the template
cx=floor(columnsT/2)+1;   cy=floor(rowsT/2)+1;

%Accumulator
accum=zeros(rows,columns);
%Template Position
for i=cx:columns-cx
  for j=cy:rows-cy
     %Template elements
       for x=1-cx:cx-1
         for y=1-cy:cy-1
             err=(double(inputimage(j+y,i+x))
             -double(template(y+cy,x+cx)))^2;
         accum(j,i)=accum(j,i)+err;
       end
     end
  end
end
```

CODE 5.1

Implementing template matching.


(a) For the gray level image


(b) For the binary image

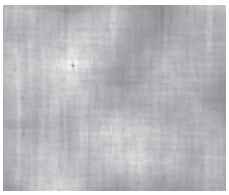

(c) For the edge image

FIGURE 5.5

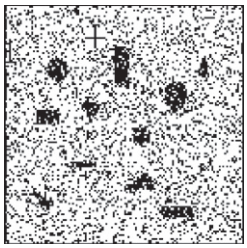
Accumulator arrays from template matching.

a result that cannot support later image interpretation that might require knowledge of more than just the best match.

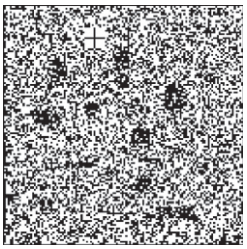
The results of applying the template matching procedure are illustrated in Figure 5.5. This example shows the accumulator arrays for matching the images shown in Figures 5.3(a), 5.4(a) and (b) with their respective templates. The dark points in each image are at the coordinates of the origin of the position where the template best matched the image (the minimum). Note that there is a border where the template has not been matched to the image data. At these border points, the template extended beyond the image data, so no matching has been performed. This is the same border as experienced with template convolution, Section 3.4.1. We can observe that a clearer minimum is obtained (Figure 5.5(c)) from the edge images of Figure 5.4. This is because for gray level and binary images, there is some match when the template is not exactly in the best position. In the case of edges, the count of matching pixels is less.

Most applications require further degrees of freedom such as rotation (orientation), scale (size), or perspective deformations. Rotation can be handled by rotating the template, or by using polar coordinates; scale invariance can be achieved using templates of differing size. Having more parameters of interest implies that the accumulator space becomes larger; its dimensions increase by one for each extra parameter of interest. **Position**-invariant template matching, as considered here, implies a 2D parameter space, whereas the extension to **scale**- and **position**-invariant template matching requires a 3D parameter space.

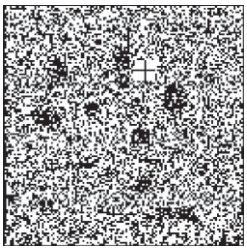
The computational cost of template matching is **large**. If the template is square and of size $m \times m$ and is matched to an image of size $N \times N$, since the m^2 pixels are matched at all image points (except for the border), the computational cost is $O(N^2m^2)$. This is the cost for position-invariant template matching. Any further parameters of interest **increase** the computational cost in proportion to the number of values of the extra parameters. This is clearly a large penalty and so a direct digital implementation of template matching is slow. Accordingly, this guarantees interest in techniques that can deliver the same result, but faster, such as using a Fourier implementation based on fast transform calculus.



(a) Extraction (of the black rectangle) in some noise



(b) Extraction in a lot of noise



(c) Extraction in too much noise (failed)

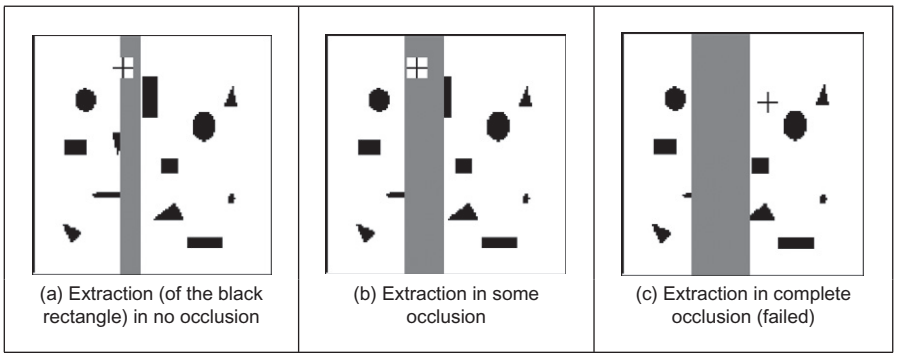
FIGURE 5.6

Template matching in noisy images.

The main **advantages** of template matching are its **insensitivity** to *noise* and *occlusion*. Noise can occur in any image, on any signal—just like on a telephone line. In digital photographs, the noise might appear low, but in computer vision it is made worse by edge detection by virtue of the differencing (differentiation) processes. Likewise, shapes can easily be occluded or **hidden**: a person can walk behind a lamp post or illumination can also cause occlusion. The **averaging** inherent in template matching reduces the susceptibility to noise; the **maximization** process reduces susceptibility to occlusion.

These advantages are illustrated in Figure 5.6 which illustrates detection in the presence of increasing noise. Here, we will use template matching to locate the region containing the vertical rectangle near the top of the image (so we are matching a binary template of a black template on a white background to the binary image). The lowest noise level is shown in Figure. 5.6(a) and the highest is shown in Figure 5.6(c); the position of the origin of the detected rectangle is shown as a black cross in a white square. The position of the origin of the region containing the rectangle is detected correctly in Figure 5.6(a) and (b) but incorrectly in the noisiest image (Figure 5.6(c)). Clearly, template matching can handle quite high noise corruption. (Admittedly this is somewhat artificial: the noise would usually be filtered out by one of the techniques described in Chapter 3, but we are illustrating basic properties here.) The ability to handle noise is shown by correct determination of the position of the target shape, until the noise becomes too much and there are more points due to noise than there are due to the shape itself. When this occurs, the votes resulting from the noise exceed those occurring from the shape, and so the maximum is not found where the shape exists.

Occlusion is shown by placing a gray bar across the image; in Figure 5.7(a), the bar does not occlude (or hide) the target rectangle, whereas in Figure 5.7(c) the rectangle is completely obscured. As with performance in the presence of noise, detection of the shape fails when the votes occurring from the shape exceed those from the rest of the image (the nonshape points), and the cross indicating



(a) Extraction (of the black rectangle) in no occlusion

(b) Extraction in some occlusion

(c) Extraction in complete occlusion (failed)

FIGURE 5.7

Template matching in occluded images.

the position of the origin of the region containing the rectangle is drawn in completely the wrong place. This is what happens when the rectangle is completely obscured in Figure 5.7(c).

So it can operate well, with practical advantage. We can include edge detection to concentrate on a shape's borders. Its main problem is still **speed**: a direct implementation is slow, especially when handling shapes that are rotated or scaled (and there are other implementation difficulties too). Recalling that from Section 3.4.2 template matching can be speeded up by using the Fourier transform, let us see if that can be used here too.

5.3.2 Fourier transform implementation

We can implement template matching via the Fourier transform by using the **duality** between convolution and multiplication, which was discussed in Section 3.4.2. This duality establishes that a multiplication in the space domain corresponds to a convolution in the frequency domain and vice versa. This can be exploited for faster computation by using the frequency domain, given the FFT algorithm. Thus, in order to find a shape, we can compute the cross-correlation as a multiplication in the frequency domain. However, the matching process in Eq. (5.11) is actually **correlation** (Section 2.3), **not** convolution. Thus, we need to express the correlation in terms of a convolution. This can be done as follows. First, we can rewrite the *correlation* (denoted by $\otimes$) in Eq. (5.11) as

$$\mathbf{I} \otimes \mathbf{T} = \sum_{(x,y) \in W} \mathbf{I}_{x',y'} \mathbf{T}_{x'-i,y'-j} \tag{5.17}$$

where $x' = x + i$ and $y' = y + j$. **Convolution** (denoted by *) is defined as

$$\mathbf{I} * \mathbf{T} = \sum_{(x,y) \in W} \mathbf{I}_{x',y'} \mathbf{T}_{i-x',j-y'} \tag{5.18}$$

Thus, in order to implement template matching in the frequency domain, we need to express Eq. (5.17) in terms of Eq. (5.18). This can be achieved by considering that

$$\mathbf{I} \otimes \mathbf{T} = \mathbf{I} * \mathbf{T}' = \sum_{(x,y) \in W} \mathbf{I}_{x',y'} \mathbf{T}'_{i-x',j-y'} \tag{5.19}$$

where

$$\mathbf{T}' = \mathbf{T}_{-x,-y} \tag{5.20}$$

That is, correlation is equivalent to convolution when the template is changed according to Eq. (5.20). This equation reverses the coordinate axes and it corresponds to a horizontal and a vertical flip.

In the frequency domain, convolution corresponds to **multiplication**. As such, Eq. (5.19) can be implemented by

$$\mathbf{I} \otimes \mathbf{T} = \mathbf{I} * \mathbf{T}' = \Im^{-1}(\Im(\mathbf{I}) \times \Im(\mathbf{T}')) \tag{5.21}$$

where $\Im$ denotes Fourier transformation as in Chapter 2 (and calculated by the FFT) and $\Im^{-1}$ denotes the inverse FFT. Note that the multiplication operator actually operates point by point, so each point is the product of the pixels at the same position in each image (in Mathcad the operation is .* and in Matlab it is called `vectorise`). This is computationally faster than its direct implementation, given the speed advantage of the FFT. There are two ways to implement this equation. In the first approach, we can compute $\mathbf{T}'$ by flipping the template and then computing its Fourier transform $\Im(\mathbf{T}')$. In the second approach, we compute the transform of $\Im(\mathbf{T})$ and then we compute its complex **conjugate**. That is,

$$\Im(\mathbf{T}') = [\Im(\mathbf{T})]^* \tag{5.22}$$

where [ ]* denotes the complex conjugate of the transform data (yes, we agree it's an unfortunate symbol clash with convolution, but they are both standard symbols). So conjugation of the transform of the template implies that the product of the two transforms leads to correlation. (Since this product is point by point, the two images/matrices need to be of the same size.) That is,

$$\mathbf{I} \otimes \mathbf{T} = \mathbf{I} * \mathbf{T}' = \Im^{-1}(\Im(\mathbf{I}) \times [\Im(\mathbf{T})]^*) \tag{5.23}$$

For both implementations, Eqs (5.21) and (5.23) will evaluate the match and more quickly for large templates than by direct implementation of template matching (as per Section 3.4.2). Note that one assumption is that the transforms are of the same size, even though the template's shape is usually much smaller than the image. There is actually a selection of approaches; a simple solution is to include extra zero values (*zero-padding*) to make the image of the template the same size as the image.

The code to implement template matching by Fourier, `FTConv`, is given in Code 5.2. The implementation takes the image and the flipped template. The

template is zero-padded and then transforms are evaluated. The required convolution is obtained by multiplying the transforms and then applying the inverse. The resulting image is the magnitude of the inverse transform. This could naturally be invoked as a single function, rather than as procedure, but the implementation is less clear. This process can be formulated using brightness or edge data, as appropriate. Should we seek **scale** invariance, to find the position of a template irrespective of its size, then we need to formulate a set of templates that range in size between the maximum and minimum expected variation. Each of the templates of differing size is then matched by frequency domain multiplication. The maximum frequency domain value, for all sizes of template, indicates the position of the template and, naturally, gives a value for its size. This can of course be a rather lengthy procedure when the template ranges considerably in size.

```
%Fourier Transform Convolution

function FTConv(inputimage,template)

%image size
[rows,columns]=size(inputimage);

%FT
Fimage=fft2(inputimage,rows,columns);
Ftemplate=fft2(template,rows,columns);

%Convolution
G=Fimage.*Ftemplate;

%Modulus
Z=log(abs(fftshift(G)));

%Inverse
R=real(ifft2(G));
```

CODE 5.2

Implementing convolution by the frequency domain.

Figure 5.8 illustrates the results of template matching in the Fourier domain using the image and template as shown in Figure 5.3. Figure 5.8(a) shows the flipped and padded template. The Fourier transforms of the image and the flipped template are given in Figure 5.8(b) and (c), respectively. These transforms are multiplied, point by point, to achieve the image in Figure 5.8(d). When this is inverse Fourier transformed, the result (Figure 5.8(e)) shows where the template best matched the image (the coordinates of the template's top left-hand corner). The result image contains several local maximum (in white). This can be

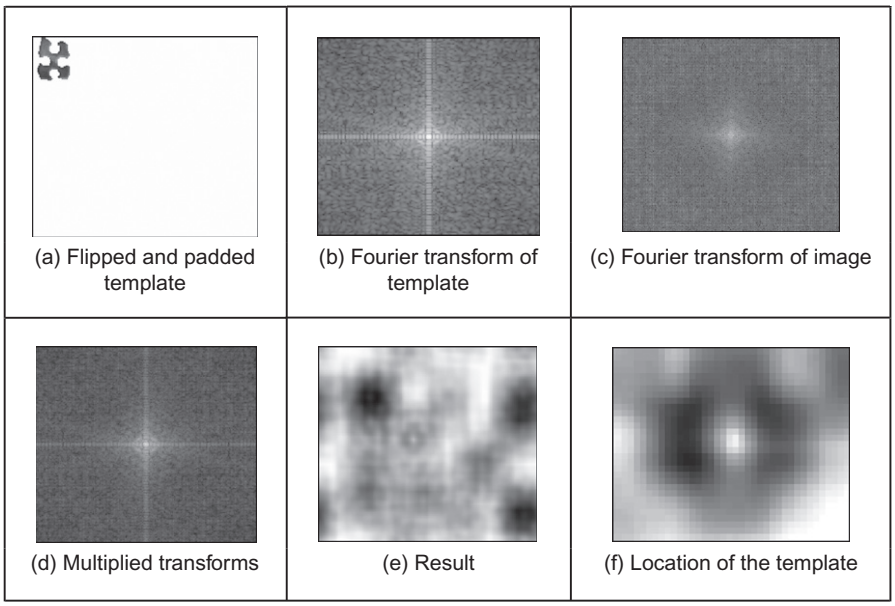


FIGURE 5.8

Template matching by Fourier transformation.

explained by the fact that this implementation does not consider the term in Eq. (5.10). Additionally, the shape can partially match several patterns in the image. Figure 5.8(f) shows a zoom of the region where the peak is located. We can see that this peak is well defined. In contrast to template matching, the implementation in the frequency domain does not have any border. This is due to the fact that Fourier theory assumes picture replication to infinity. Note that in application, the Fourier transforms do not need to be rearranged (`fftshif`) so that the d.c. is at the center, since this has been done here for display purposes only.

There are several further difficulties in using the transform domain for template matching in discrete images. If we seek rotation invariance, then an image can be expressed in terms of its polar coordinates. Discretization gives further difficulty since the points in a rotated discrete shape can map imperfectly to the original shape. This problem is better manifest when an image is scaled in size to become larger. In such a case, the spacing between points will increase in the enlarged image. The difficulty is how to allocate values for pixels in the enlarged image which are not defined in the enlargement process. There are several interpolation approaches, but it can often appear prudent to reformulate the original approach. Further difficulties can include the influence of the image borders: Fourier theory assumes that an image replicates spatially to infinity. Such difficulty can be reduced by using window operators, such as the Hamming or the Hanning windows. These difficulties do not obtain for optical Fourier transforms

and so using the Fourier transform for position-invariant template matching is often confined to optical implementations.

5.3.3 Discussion of template matching

The advantages associated with template matching are mainly theoretical since it can be very difficult to develop a template matching technique that operates satisfactorily. The results presented here have been for **position** invariance only. This can cause difficulty if invariance to **rotation** and **scale** is also required. This is because the template is stored as a discrete set of points. When these are rotated, **gaps** can appear due to the discrete nature of the coordinate system. If the template is increased in size then again there will be missing points in the scaled-up version. Again, there is a frequency domain version that can handle variation in size, since scale-invariant template matching can be achieved using the *Mellin transform* (Bracewell, 1986). This avoids using many templates to accommodate the variation in size by evaluating the scale-invariant match in a single pass. The Mellin transform essentially scales the spatial coordinates of the image using an exponential function. A point is then moved to a position given by a logarithmic function of its original coordinates. The transform of the scaled image is then multiplied by the transform of the template. The maximum again indicates the best match between the transform and the image. This can be considered to be equivalent to a change of variable. The logarithmic mapping ensures that scaling (multiplication) becomes addition. By the logarithmic mapping, the problem of scale invariance becomes a problem of finding the position of a match.

The Mellin transform only provides scale-invariant matching. For scale and position invariance, the Mellin transform is combined with the Fourier transform, to give the *Fourier–Mellin* transform. The Fourier–Mellin transform has many disadvantages in a digital implementation due to the problems in spatial resolution though there are approaches to reduce these problems (Altman and Reitbock, 1984), as well as the difficulties with discrete images experienced in Fourier transform approaches.

Again, the Mellin transform appears to be much better suited to an **optical** implementation (Casasent and Psaltis, 1977), where **continuous** functions are available, rather than to discrete image analysis. A further difficulty with the Mellin transform is that its result is independent of the **form factor** of the template. Accordingly, a rectangle and a square appear to be the same to this transform. This implies a loss of information since the form factor can indicate that an object has been imaged from an oblique angle. There is actually resurgent interest in *log-polar mappings* for image analysis (e.g., Traver and Pla, 2003; Zokai and Wolberg, 2005).

So there are innate difficulties with template matching whether it is implemented directly or by transform operations. For these reasons, and because many shape extraction techniques require more than just edge or brightness data, direct digital implementations of feature extraction are usually preferred. This is perhaps

also influenced by the speed advantage that one popular technique can confer over template matching. This is the Hough transform, which is covered in Section 5.5. Before that, we shall consider techniques which consider object extraction by collections of low-level features. These can avoid the computational requirements of template matching by treating shapes as collections of features.

5.4 Feature extraction by low-level features

There have been many approaches to feature extraction which combine a variety of features. It is possible to characterize objects by measures that we have already developed, by low-level features, local features (such edges and corners), and by global features (such as color). Later we shall find these can be grouped to give structure or shape (in this chapter and the next), and appearance (called texture, Chapter 8). The drivers for the earlier approaches which combine low-level features are the need to be able to search databases for particular images. This is known as image retrieval, and in content-based retrieval, which uses techniques from image processing and computer vision, there are approaches which combine a selection of features (Smeulders et al., 2000). Alternative search strategies include using text or sketches and these are not of interest in the domain of this book. More recently, the trend is to develop features which include and target human descriptions and use techniques from machine intelligence (Datta et al., 2008), which also implies understanding of semantics (how people describe images) as compared with the results of automated image analysis.

There is also interest in recognizing objects, and hence images, by collecting descriptors for local features (Mikolajczyk and Schmid, 2005). These can find application not just in image retrieval but also in **stereo** computer vision, **navigating robots** by computer vision and when **stitching** together multiple images to build a much larger **panorama** image. Much of this material relates to whole applications and therefore can rely not just on collecting local features, shape, texture but also on classification. In these respects in this chapter, we shall provide coverage of some of the basic ways to combine low-level feature descriptions. Essentially, these approaches show how techniques that have already been covered can be combined in such a way as to achieve a description by which an object can be recognized. The approaches tend to rely on the use of machine learning approaches to determine the relevant data (top filter it so as to understand its structure), so the approaches are described in basis only here and the classification approaches are described later in Chapter 8.

5.4.1 Appearance-based approaches

5.4.1.1 Object detection by templates

The *Viola–Jones approach* essentially uses the form of Haar wavelets defined in Section 2.7.3.2 as a basis for object detection (Viola and Jones, 2001) which was

(a) Face image

(b) Template for eyes and nose bridge

(c) Best match or template (b) to image (a)

FIGURE 5.9

Object extraction by Haar wavelet-based features.

later extended to be one of the most popular techniques for detecting human faces in images (Viola and Jones, 2004). Using rectangles to detect image features is an approximation, as there are features which can describe curved structure (derived using Gabor wavelet for example). It is however a fast approximation, since the features can be detected using the **integral image** approach. If we are to consider the face image in Figure 5.9(a), then the eyes are darker than the cheeks which are immediately below them, and the eyes are also darker than the bridge of the nose. As such, if we match the template in Figure 5.9(b) (this is the inverted form of the template in Figure 2.29(c)), then superimposing this template on the image at the position where it best matches the face leads to the image of Figure 5.9(c). The result is not too surprising, since it finds two dark parts between which there is a light part, and only the eyes and the bridge of the nose fit this description. (We could of course have a nostril template but (a) you might be eating your dinner and (b) when you look closely, quite a lot of the image fits the description "two small dark blobs with a light bit in the middle"—we can successfully find the eyes since they are a large structure fitting the template well.)

In this way we can define a series of templates (those in Figure 2.29) and match them to the image. In this way we can find the underlying shape. We need to sort the results to determine which are the most important and which collection best describes the face. That is where the approach advances to machine learning, which comes later in Chapter 8. For now, we rank the filters as to their importance and then find shapes by using a collection of these low-level features. The original technique was phrased around detecting objects (Viola and Jones, 2001) and later phrased around finding human faces in particular (Viola and Jones, 2004), and it has now become one of the stock approaches to detecting faces automatically within image data.

There are limitations to this approach, naturally. The use of rectangular features allows fast calculation but does not match well with structures which have a smoother contour. There are very many features possible in templates of any reasonable size, and so the set of features must be pruned so that the best are selected, and that is where the machine learning processes are necessary. In turn

this implies that the feature extraction process needs training (in features and in data)—and that is similar indeed to human vision. There are demonstration versions of the technique and improvements include the use of rotated Haar features (Lienhart et al., 2003) as well as inspiring many of the more recent approaches which collect parts for recognition.

5.4.1.2 Object detection by combinations of parts

There have been many approaches which apply **wavelets**, and ones which are more complex than Haar wavelets, to detect objects by combinations of parts. These approaches allow for greater flexibility in the representation of the part since the wavelet can capture frequency, orientation, and position (thus incurring the cost of computational complexity). A major advantage is that scale can be used, and objects can exist at, or persist over, a selection of scales. One such approach used wavelets as a basis for detecting people and cars (Schneiderman and Kanade, 2004) and even a door handle, thus emphasizing generality of the approach. As with the Viola–Jones approach, this method requires deployment of machine learning techniques which then involves training. In this method, the training occurs over different viewpoints to factor out the subject's—or object's—pose. The method groups input data into sets, and each set is a part. For a human face, the parts include the eyes, nose, and mouth, and some unnamed but classified face regions, and these parts are (statistically) interdependent in most natural objects. Then machine learning techniques are used to maximize the likelihood of finding the parts correctly. Highly impressive results have been provided, though again the performance of the technique depends on training as well as on other factors. The main point of the technique here is that wavelets can allow for greater freedom when representing an object as a collection of parts.

In our own research we have used Gabor wavelets in ear biometrics, where we can recognize a person's identity by analysis of the appearance of the ear (Hurley et al., 2008). It might be the ugliest biometric, but it also appears the most immune to effects of aging: ears are fully formed at birth and change little throughout life, unlike the human face which changes rapidly as children grow teeth and then the general decline includes wrinkles and a few sags (unless a surgeon's expertise is deployed). In a way ears are like fingerprints, but the features are less clear. In our own research in biometrics, we have used Gabor wavelets to capture the ear's features (Arbab-Zavar and Nixon, 2011) in particular those relating to smooth curves. To achieve rotational invariance (in case a subject's head was tilted when the image was acquired), a radial scan was taken based on an ear's center point (Figure 5.10(a)) deriving the two transformed regions in Figure 5.10(b) and (d) which are the same region for different images of the same ear. Then, these regions are transformed using a Gabor wavelet approach for which the real parts of the transform at two scales are shown in Figure 5.10(c) and (e). Here, the detail is preserved at the short wavelength and the larger structures are detected at longer wavelengths. In both cases, the prominent smooth

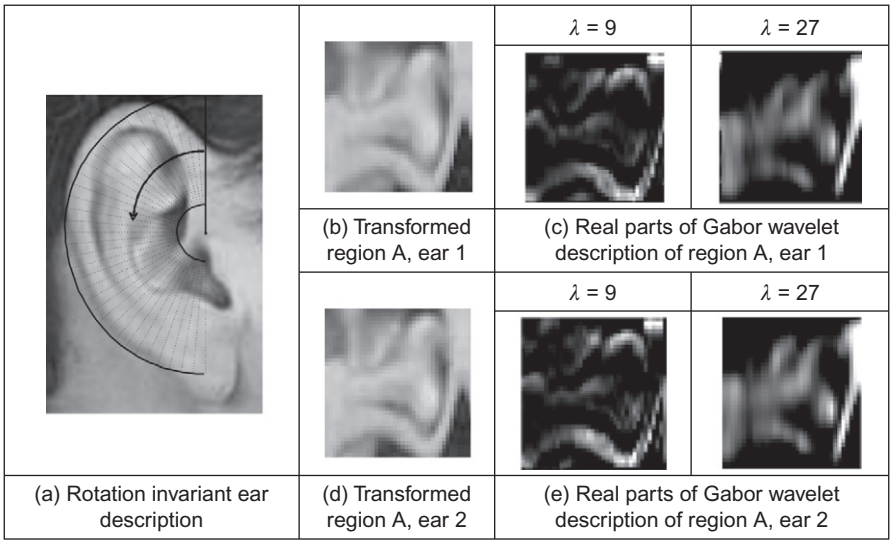


FIGURE 5.10

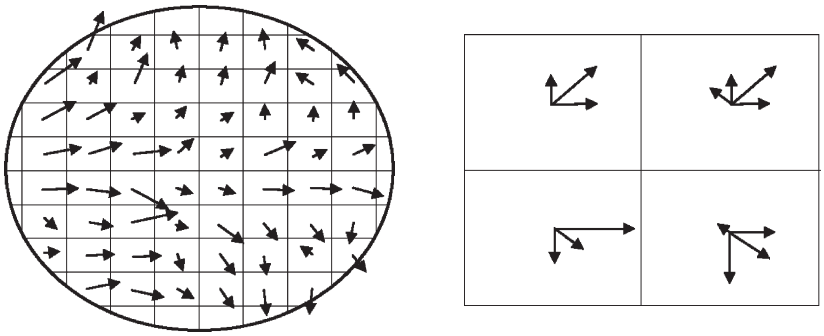
Applying Gabor wavelets in ear biometrics (Arbab-Zavar and Nixon, 2011).

structures are captured by the technique, leading to successful recognition of the subjects.

5.4.2 Distribution-based descriptors

5.4.2.1 Description by interest points

Lowe's SIFT (Lowe, 2004), Section 4.4.2.1, actually combines a scale-invariant region **detector** with a **descriptor** which is based on the gradient distribution in the detected regions. The approach not only detects interest points but also provides a description for recognition purposes. The descriptor is represented by a 3D histogram of gradient locations and orientations and is created by first computing the gradient magnitude and orientation at each image point within the 8×8 region around the keypoint location, as shown in Figure 5.11. These values are weighted by a Gaussian windowing function, indicated by the overlaid circle in Figure 5.11(a) wherein the standard deviation is chosen according to the number of samples in the region (its width). This avoids fluctuation in the description with differing values of the keypoint's location and emphasizes less the gradients that are far from the center. These samples are then accumulated into orientation histograms summarizing the contents of the four 4×4 subregions, as shown in Figure 5.11(b), with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This involves a binning procedure as the histogram is quantized into a smaller number of levels (here



(a) Image gradients (b) Keypoint descriptor

FIGURE 5.11

SIFT keypoint descriptor (Lowe, 2004).

eight compass directions are shown). The descriptor is then a vector of the magnitudes of the elements at each compass direction and in this case has $4 \times 8 = 32$ elements. This figure shows a 2×2 descriptor array derived from an 8×8 set of samples and other arrangements are possible, such as 4×4 descriptors derived from a 16×16 sample array giving a 128 element descriptor. The final stage is to normalize the magnitudes, so the description is illumination invariant. Given that SIFT has detected the set of keypoints and we have descriptions attached to each of those keypoints, we can then describe a shape by using the collection of parts detected by the SIFT technique. There is a variety of parameters that can be chosen within the approach, and the optimization process is ably described (Lowe, 2004) along with demonstration that the technique can be used to recognize objects, even in the presence of clutter and occlusion.

The *SURF descriptor* (Bay et al., 2008), Section 4.4.2.2, describes the distribution of the intensity content within the interest point neighborhood, similar to SIFT (both approaches combine detection with description). In SURF, first a square region is constructed which is centered on an interest point and oriented along the detected orientation (detected via the Haar wavelets). Then, the description is derived from the Haar wavelet responses within the sub-windows and the approach argues the approach "reduces the time for feature computation and matching and has proven to simultaneously increase the robustness."

The major performance evaluation (Mikolajczyk and Schmid, 2005) compared the performance of descriptors computed for local interest regions and studied a number of operators, concerning in particular the effects of geometric and affine transformations, for matching and recognition of the same object or scene. The operators included a form of Gabor wavelets and SIFT (and some operators we have yet to encounter in this text), and also introduced the *gradient location and orientation histogram (GLOH)* which is an extension of the SIFT descriptor, and

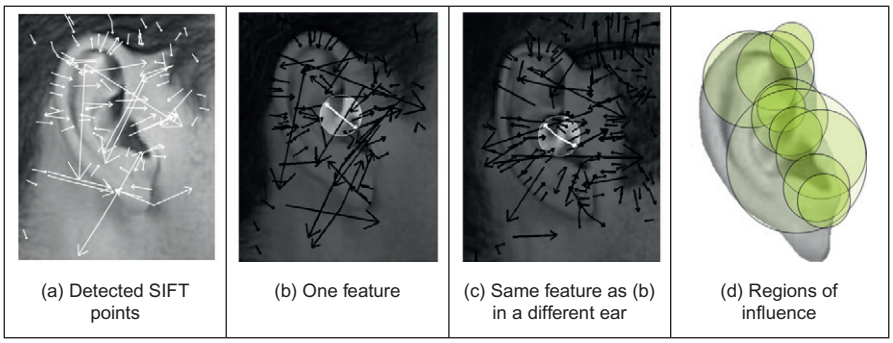


FIGURE 5.12

Applying SIFT in ear biometrics (Arbab-Zavar and Nixon, 2011).

which appeared to offer better performance. The survey predated SURF and so it was not included. SIFT also performed well and there have been many applications of the SIFT approach for recognizing objects in images, and the applications of SURF are burgeoning. One approach aimed to determine those key frames and shots of a video containing a particular object with ease and convenience of the Google search engine (Sivic et al., 2003). In this approach, elliptical regions are represented by a 128-dimensional vector using the SIFT descriptor which was chosen by virtue of superior performance, especially when the object's positions could vary by small amounts. From this, descriptions are constructed using machine learning techniques.

In common with other object recognition approaches, we have deployed SIFT for ear biometrics (Bustard and Nixon, 2010; Arbab-Zavar and Nixon, 2011) to capture the description of an individual's ear by a constellation of ear parts, again confirming that people appear unique by their ear. Here, the points detected are those which are significant across scales and thus provide an alternative characterization (to the earlier Gabor wavelet analysis in Section 5.4.1.2) of the ear's appearance. Figure 5.12(a) shows the SIFT points detected within a human ear and Figure 5.12(b) and (c) shows the same point (the crus of helix, no less) being detected in two different ears, and Figure 5.12(d) shows the domains of the SIFT points dominant in the ear biometrics procedure. Note that these points do not include the outer perimeter of the ear, which was described by Gabor wavelets. Recognition by the SIFT features was complemented by the Gabor features, as we derive descriptions of different regions, leading to the successful identification of the subjects by their ears. An extended discussion of how ears can be used as a biometric and the range of techniques that can be used for recognition is available (Hurley et al., 2008).

As such we have concerned a topical area of major current interest. Note that one survey on interest point detectors (Tuytelaars and Mikolajczyk, 2007) noted

"the repeatability of the local feature detectors is still very limited, with repeatability scores below 50% being quite common" and this will naturally affect discriminative capability. However, there are now many studies deploying interest point techniques for image matching, which show considerable performance capability. It is likely that the performance will be improved by technique refinement and analysis, and therefore performance comparison and abilities will continue to develop.

5.4.2.2 Characterizing object appearance and shape

There has long been an interest in detecting pedestrians within scenes, more for automated surveillance analysis than for biometric purposes. The techniques have included use of Haar features and more recently the SIFT description. An approach called the *histogram of oriented gradients (HoG)* (Dalal and Triggs, 2005) has been receiving much interest. This captures edge or gradient structure that is very characteristic of local shape in a way which is relatively unaffected by appearance changes. Essentially, it forms a template and deploys machine learning approaches to expedite recognition, in an effective way. In this way it is an extension to describing objects by a histogram of the edge gradients.

First, edges are detected by the improved first-order detector as shown in Figure 4.4 and an edge image is created. Then, a vote is determined from a pixel's edge magnitude and direction and stored in a histogram. The direction is "binned" in that votes are cast into roughly quantized histogram ranges and these votes are derived from cells, which group neighborhoods of pixels. One implementation is to use 8×8 image cells and to group these into 20° ranges (thus nine ranges within 180° of unsigned edge direction). Local contrast normalization is used to handle variation in gradient magnitude due to change in illumination and contrast with the background, and this was determined to be an important stage. This normalization is applied in blocks, eventually leading to the person's description which can then be learned by using machine learning approaches. Naturally there is a gamut of choices to be made, such as the choice of edge detection operator, inclusion of operator, cell size, the number of bins in the histogram, and use of full 360° edge direction. Robustness is achieved in that noise or other effects should not change the histograms much: the filtering is done at the description stage rather than at the image stage (as with wavelet-based approaches).

The process of building the HoG description is illustrated in Figure 5.13 where (a) is the original image; (b) is the gradient magnitude constructed from the absolute values of the improved first-order difference operator; (c) is the grid of 8×8, superimposed on the edge direction image; and (d) illustrates the 3×3 (rectangular) grouping of the cells, superimposed on the histograms of gradient data. There is a rather natural balance between the grid size and the size of the grouping arrangements, though these can be investigated in application. Components of the

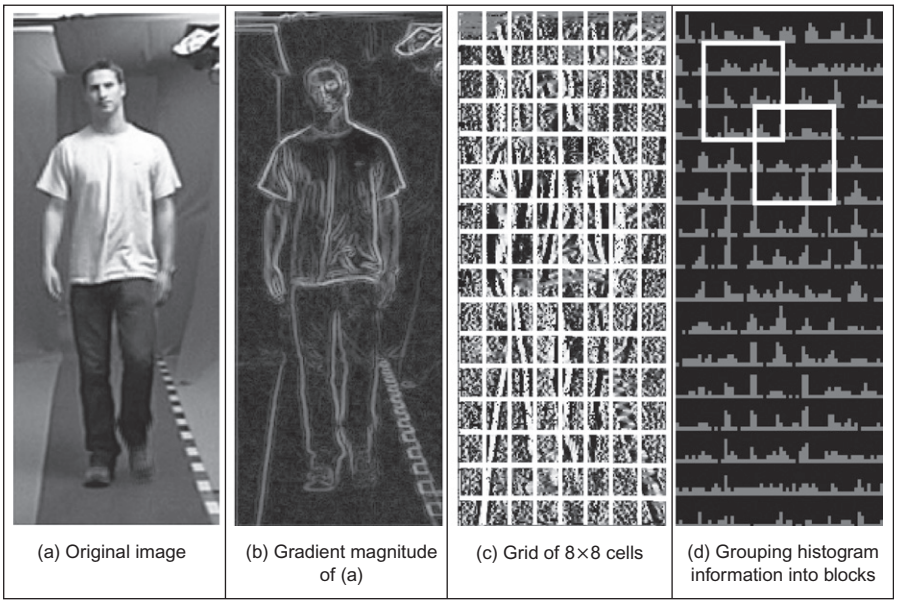


FIGURE 5.13

Illustrating the HoG description.

walking person can be seen especially in the preponderance of vertical edge components in the legs and thorax. The grouping and normalization of these data lead to the descriptor which can be deployed so as to detect humans/pedestrians in static images.

The approach is not restricted to detecting pedestrians since it can be trained to detect different shapes and it has been applied elsewhere. Given there is much interest in speed of computation, rather unexpectedly a Fast HoG was to appear soon after the original HoG (Zhu et al., 2006) and which claims 30 fps capability. An alternative approach and one which confers greater generality—especially with humans—is to include the possibility of deformation, as will be covered in Section 6.2.

Essentially, these approaches can achieve fast extraction by decomposing a shape into its constituent parts. Clearly one detraction of the techniques is that if you are to change implementation—or to detect other objects—then this requires construction of the necessary models and parts, and that can be quite demanding. If fact, it can be less demanding to include shape, and as template matching can give a guaranteed result, another class of approaches is to reformulate template matching so as to improve speed, i.e., the Hough transform explained in the following section.

5.5 Hough transform

5.5.1 Overview

The *Hough transform* (HT) (Hough, 1962) is a technique that locates shapes in images. In particular, it has been used to extract **lines**, **circles**, and **ellipses** (or conic sections). In the case of lines, its mathematical definition is equivalent to the Radon transform (Deans, 1981). The HT was introduced by Hough (1962) and then used to find bubble tracks rather than shapes in images. However, Rosenfeld noted its potential advantages as an image processing algorithm (Rosenfeld, 1969). The HT was thus implemented to find lines in images (Duda and Hart, 1972) and it has been extended greatly, since it has many advantages and many potential routes for improvement. Its prime advantage is that it can deliver the **same** result as that for template matching, but **faster** (Stockman and Agrawala, 1977; Sklansky, 1978; Princen et al., 1992b). This is achieved by a reformulation of the template matching process, based on an *evidence-gathering* approach where the evidence is the **votes** cast in an accumulator array. The HT implementation defines a **mapping** from the image points into an accumulator space (Hough space). The mapping is achieved in a computationally efficient manner, based on the function that describes the target shape. This mapping requires much less computational resources than template matching. However, it still requires significant storage and high computational requirements. These problems are addressed later, since they give focus for the continuing development of the HT. However, the fact that the HT is equivalent to template matching has given sufficient impetus for the technique to be among the most popular of all existing shape extraction techniques.

5.5.2 Lines

We will first consider finding lines in an image. In a Cartesian parameterization, collinear points in an image with coordinates (x,y) are related by their slope m and an intercept c according to

$$y = mx + c \tag{5.24}$$

This equation can be written in homogeneous form as

$$Ay + Bx + 1 = 0 \tag{5.25}$$

where $A = -1/c$ and $B = m/c$. Thus, a line is defined by giving a pair of values (A,B). However, we can observe a symmetry in the definition in Eq. (5.25). This equation is symmetric since a pair of coordinates (x,y) also defines a line in the space with parameters (A,B). That is, Eq. (5.25) can be seen as the equation of a line for fixed coordinates (x,y) or as the equation of a line for fixed parameters (A,B). Thus, pairs can be used to define points and lines simultaneously (Aguado et al., 2000a). The HT gathers evidence of the point (A,B) by considering that all

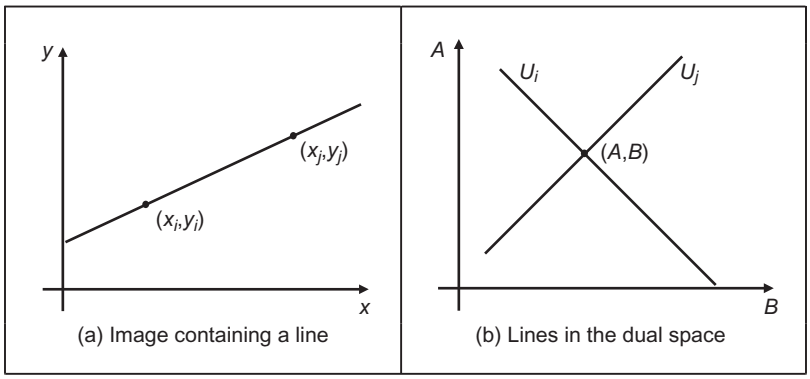


FIGURE 5.14

Illustrating the HT for lines.

the points (x,y) define the same line in the space (A,B). That is, if the set of collinear points $\{(x_i,y_i)\}$ defines the line (A,B), then

$$Ay_i + Bx_i + 1 = 0 \tag{5.26}$$

This equation can be seen as a system of equations and it can simply be rewritten in terms of the Cartesian parameterization as

$$c = -x_i m + y_i \tag{5.27}$$

Thus, to determine the line, we must find the values of the parameters (m,c) (or (A,B) in homogeneous form) that satisfy Eq. (5.27) (or Eq. (5.26), respectively). However, we must note that the system is generally overdetermined. That is, we have more equations than unknowns. Thus, we must find the solution that comes close to satisfying all the equations simultaneously. This kind of problem can be solved, for example, using linear least-squares techniques. The HT uses an evidence-gathering approach to provide the solution.

The relationship between a point (x_i,y_i) in an image and the line given in Eq. (5.27) is illustrated in Figure 5.14. The points (x_i,y_i) and (x_j,y_j) in Figure 5.14(a) define the lines U_i and U_j in Figure 5.14(b), respectively. All the collinear elements in an image will define dual lines with the same concurrent point (A,B). This is independent of the line parameterization used. The HT solves it in an efficient way by simply counting the potential solutions in an accumulator array that stores the evidence or votes. The count is made by tracing all the dual lines for each point (x_i,y_i). Each point in the trace increments an element in the array, thus the problem of line extraction is transformed in the problem of locating a maximum in the accumulator space. This strategy is robust and has demonstrated to be able to handle noise and occlusion.

The axes in the dual space represent the parameters of the line. In the case of the Cartesian parameterization, m can actually take an **infinite** range of values,

since lines can vary from horizontal to vertical. Since votes are gathered in a discrete array, this will produce **bias** errors. It is possible to consider a range of votes in the accumulator space that cover all possible values. This corresponds to techniques of antialiasing and can improve the gathering strategy (Brown, 1983; Kiryati and Bruckstein, 1991).

The implementation of the HT for lines, `HTLine`, is given in Code 5.3. It is important to observe that Eq. (5.27) is not suitable for implementation since the parameters can take an infinite range of values. In order to handle the infinite range for c, we use two arrays in the implementation in Code 5.3. When the slope m is between -45° and 45°, then c does not take a large value. For other values of m, the intercept c can take a very large value. Thus, we consider an accumulator for each case. In the second case, we use an array that stores the intercept with the x axis. This only solves the problem partially since we cannot guarantee that the value of c will be small when the slope m is between -45° and 45°.

```
%Hough Transform for Lines
function HTLine(inputimage)
%image size
[rows,columns]=size(inputimage);
%accumulator
acc1=zeros(rows,91);
acc2=zeros(columns,91);
%image
for x=1:columns
  for y=1:rows
     if(inputimage(y,x)==0)
      for m=-45:45
        b=round(y-tan((m*pi)/180)*x);
        if(b<rows & b>0)
           acc1(b,m+45+1)=acc1(b,m+45+1)+1;
        end
      end
      for m=45:135
        b=round(x-y/tan((m*pi)/180));
        if(b<columns & b>0)
          acc2(b,m-45+1)=acc2(b,m-45+1)+1;
        end
      end
    end
  end
end
```

CODE 5.3

Implementing the HT for lines.

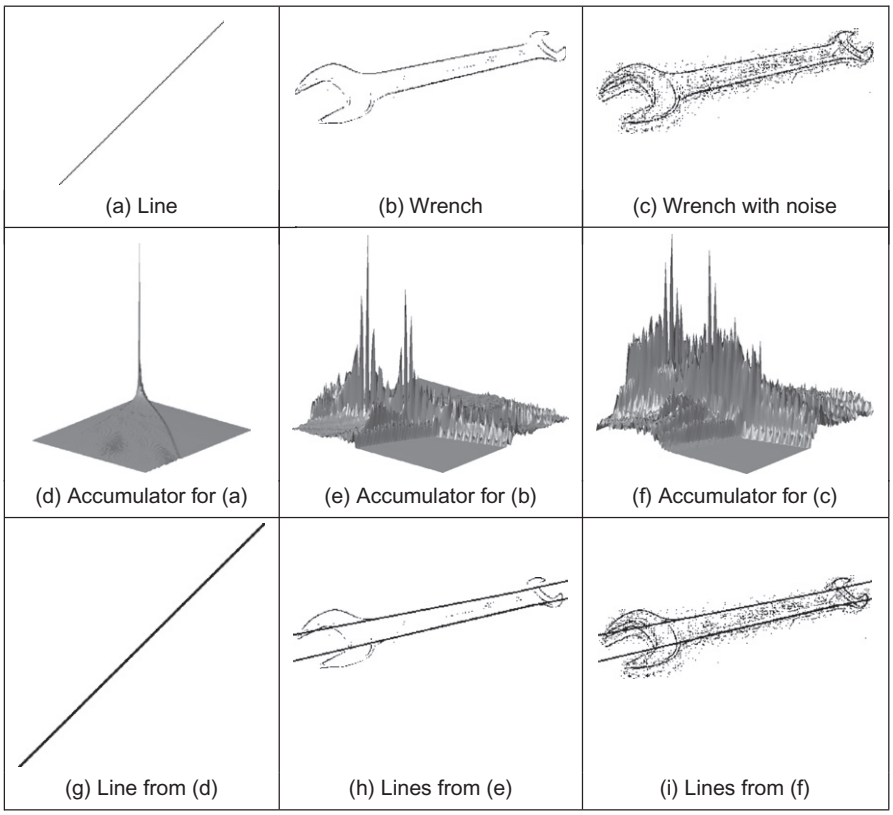


FIGURE 5.15

Applying the HT for lines.

Figure 5.15 shows three examples of locating lines using the HT implemented in Code 5.3. In Figure 5.15(a), there is a single line which generates the peak seen in Figure 5.15(d). The magnitude of the peak is proportional to the number of pixels in the line from which it was generated. The edges of the wrench in Figure 5.15(b) and (c) define two main lines. The image in Figure 5.15(c) contains much more noise. This image was obtained by using a lower threshold value in the edge detector operator which gave rise to more noise. The accumulator results of the HT for the images in Figure 5.15(b) and (c) are shown in Figure 5.15(e) and (f), respectively. We can observe the two accumulator arrays are broadly similar in shape, and that the peak in each is at the same place. The coordinates of the peaks are at combinations of parameters of the lines that best fit the image. The extra number of edge points in the noisy image of the wrench gives rise to more votes in the accumulator space, as can be seen by the increased number of votes in Figure 5.15(f) compared with Figure 5.15(e). Since the peak is in the same place, this shows that the HT can indeed tolerate noise. The results of extraction, when superimposed on

the edge image, are shown in Figure 5.15(g)–(i). Only the two lines corresponding to significant peaks have been drawn for the image of the wrench. Here, we can see that the parameters describing the lines have been extracted well. Note that the end points of the lines are not delivered by the HT, only the parameters that describe them. You have to go back to the image to obtain line length.

We can see that the HT delivers a correct response, correct estimates of the parameters used to specify the line, so long as the number of collinear points along that line exceeds the number of collinear points on any other line in the image. As such, the HT has the same properties in respect of noise and occlusion, as with template matching. However, the nonlinearity of the parameters and the discretization produce noisy accumulators. A major problem in implementing the basic HT for lines is the definition of an appropriate accumulator space. In application, Bresenham's line drawing algorithm (Bresenham, 1965) can be used to draw the lines of votes in the accumulator space. This ensures that lines of connected votes are drawn as opposed to use of Eq. (5.27) that can lead to gaps in the drawn line. Also, *backmapping* (Gerig and Klein, 1986) can be used to determine exactly which edge points contributed to a particular peak. Backmapping is an **inverse** mapping from the accumulator space to the edge data and can allow for shape analysis of the image by removal of the edge points which contributed to particular peaks, and then by reaccumulation using the HT. Note that the computational cost of the HT depends on the number of edge points (n_e) and the length of the lines formed in the parameter space (l), giving a computational cost of $O(n_e l)$. This is considerably less than that for template matching, given earlier as $O(N^2 m^2)$.

One way to avoid the problems of the Cartesian parameterization in the HT is to base the mapping function on an alternative parameterization. One of the most proven techniques is called the *foot-of-normal* parameterization. This parameterizes a line by considering a point (x,y) as a function of an angle normal to the line, passing through the origin of the image. This gives a form of the HT for lines known as the *polar HT for lines* (Duda and Hart, 1972). The point where this line intersects the line in the image is given by

$$\rho = x \cos(\theta) + y \sin(\theta) \tag{5.28}$$

where θ is the angle of the line normal to the line in an image and ρ is the length between the origin and the point where the lines intersect, as illustrated in Figure 5.16.

By recalling that two lines are perpendicular if the product of their slopes is -1 and by considering the geometry of the arrangement in Figure 5.16, we obtain

$$c = \frac{\rho}{\sin(\theta)}; \quad m = -\frac{1}{\tan(\theta)} \tag{5.29}$$

By substitution in Eq. (5.24), we obtain the polar form, Eq. (5.28). This provides a different mapping function: votes are now cast in a sinusoidal manner, in a 2D accumulator array in terms of θ and ρ, the parameters of interest. The advantage of this alternative mapping is that the values of the parameters θ and ρ are now bounded to lie within a specific range. The range for θ is within 180°; the possible values of ρ are given by the image size, since the maximum length

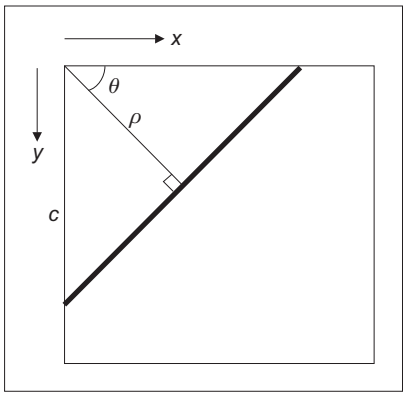


FIGURE 5.16

Polar consideration of a line.

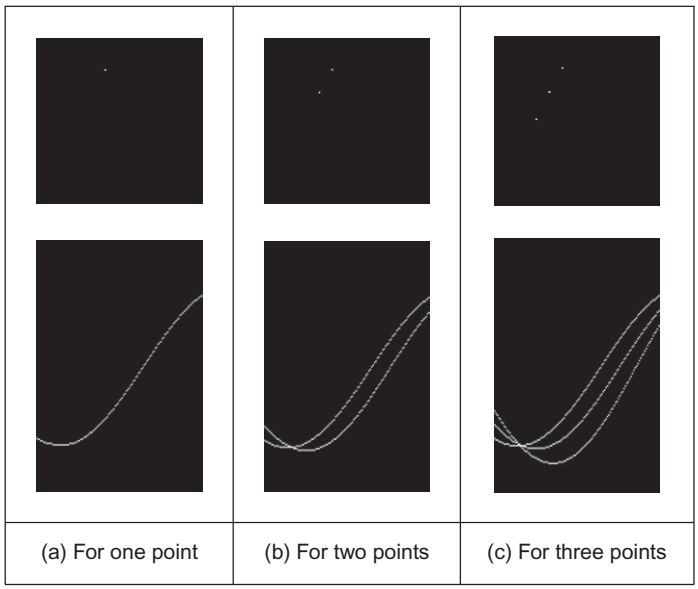


FIGURE 5.17

Images and the accumulator space of the polar HT.

of the line is $\sqrt{2} \times N$, where N is the (square) image size. The range of possible values is now fixed, so the technique is practicable.

As the voting function has now changed, we shall draw different loci in the accumulator space. In the conventional HT for lines, a straight line is mapped to a straight line as shown in Figure 5.14. In the polar HT for lines, points map to curves in the accumulator space. This is illustrated in Figure 5.17 which shows the polar HT accumulator spaces for (a) one, (b) two, and (c) three points, respectively.

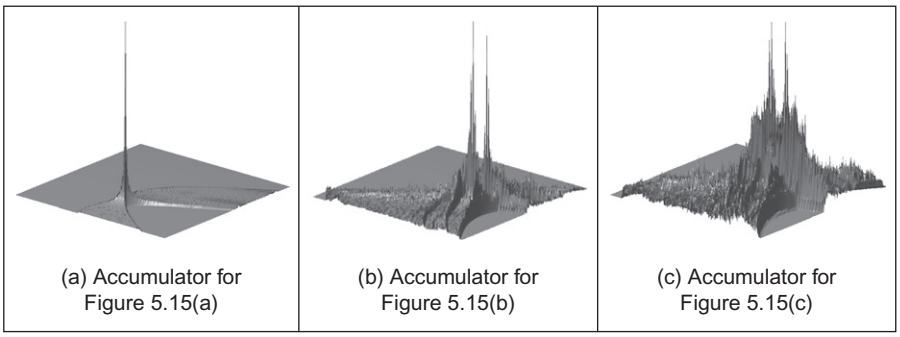
For a single point in the upper row of Figure 5.17(a), we obtain a single curve shown in the lower row of Figure 5.17(a). For two points we obtain two curves, which intersect at a position which describes the parameters of the line joining them (Figure 5.17(b)). An additional curve obtains for the third point and there is now a peak in the accumulator array containing three votes (Figure 5.17(c)).

The implementation of the **polar HT for lines** is the function `HTPLine` in Code 5.4. The accumulator array is a set of 180 bins for value of θ in the range $0-180^\circ$, and for values of ρ in the range 0 to $\sqrt{N^2+M^2}$, where $N \times M$ is the picture size. Then, for image (edge) points greater than a chosen threshold, the angle relating to the bin size is evaluated (as radians in the range $0-\pi$) and then the value of ρ is evaluated from Eq. (5.28), and the appropriate accumulator cell is incremented so long as the parameters are within range. The accumulator arrays obtained by applying this implementation to the images in Figure 5.15 are shown in Figure 5.18. Figure 5.18(a) shows that a single line defines a well-delineated peak. Figure 5.18(b) and (c) shows a clearer peak compared to the implementation of the Cartesian parameterization. This is because discretization effects are reduced in the polar parameterization. This feature makes the polar implementation far more practicable than the earlier, Cartesian, version.

```
%Polar Hough Transform for Lines

function HTPLine(inputimage)

%image size
[rows,columns]=size(inputimage);

%accumulator
rmax=round(sqrt(rows^2+columns^2));
acc=zeros(rmax,180);

%image
for x=1:columns
  for y=1:rows
     if(inputimage(y,x)==0)
        for m=1:180
           r=round(x*cos((m*pi)/180)
             +y*sin((m*pi)/180));
           if(r<rmax & r>0)
              acc(r,m)= acc(r,m)+1; end
        end
     end
  end
end
```

CODE 5.4

Implementation of the polar HT for lines.



(a) Accumulator for Figure 5.15(a)

(b) Accumulator for Figure 5.15(b)

(c) Accumulator for Figure 5.15(c)

FIGURE 5.18

Applying the polar HT for lines.

5.5.3 HT for circles

The HT can be extended by replacing the equation of the curve in the detection process. The equation of the curve can be given in **explicit** or **parametric** form. In explicit form, the HT can be defined by considering the equation for a circle given by

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{5.30}$$

This equation defines a locus of points (x,y) centered on an origin (x_0,y_0) and with radius r. This equation can again be visualized in two dual ways: as a locus of points (x,y) in an image and as a locus of points (x_0,y_0) centered on (x,y) with radius r.

Figure 5.19 illustrates this dual definition. Each edge point in Figure 5.19(a) defines a set of circles in the accumulator space. These circles are defined by all possible values of the radius and they are centered on the coordinates of the edge point. Figure 5.19(b) shows three circles defined by three edge points. These circles are defined for a given radius value. Actually, each edge point defines circles for the other values of the radius. This implies that the accumulator space is 3D (for the three parameters of interest) and that edge points map to a **cone** of votes in the accumulator space. Figure 5.19(c) illustrates this accumulator. After gathering evidence of all the edge points, the maximum in the accumulator space again corresponds to the parameters of the circle in the original image. The procedure of evidence gathering is the same as that for the HT for lines, but votes are generated in cones, according to Eq. (5.30).

Equation (5.30) can be defined in **parametric** form as

$$x = x_0 + r\cos(\theta); \quad y = y_0 + r\sin(\theta) \tag{5.31}$$

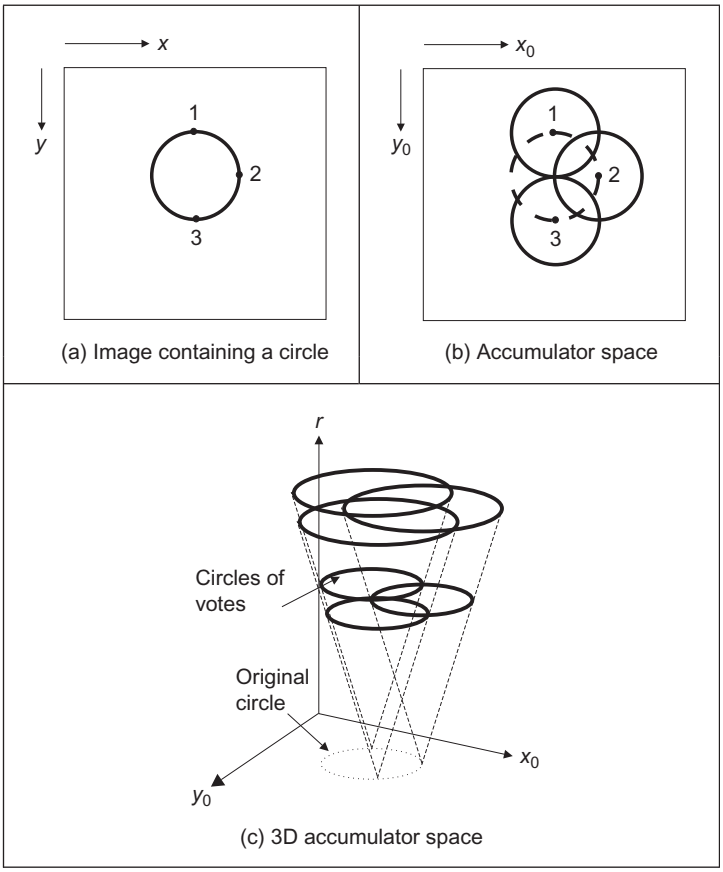


FIGURE 5.19

Illustrating the HT for circles.

The advantage of this representation is that it allows us to solve for the parameters. Thus, the HT mapping is defined by

$$x_0 = x - r\cos(\theta); \quad y_0 = y - r\sin(\theta) \tag{5.32}$$

These equations define the points in the accumulator space (Figure 5.19(b)) dependent on the radius r. Note that θ is not a free parameter but defines the trace of the curve. The trace of the curve (or surface) is commonly referred to as the **point spread function**.

The implementation of the HT for circles, `HTCircle`, is shown in Code 5.5. This is similar to the HT for lines, except that the voting function corresponds to that in Eq. (5.32) and the accumulator space is for circle data. The accumulator in the implementation is actually 2D, in terms of the center parameters for a fixed

value of the radius given as an argument to the function. This function should be called for all potential radii. A circle of votes is generated by varying t (i.e., θ, but Matlab does not allow Greek symbols!) from 0° to 360°. The discretization of t controls the granularity of voting, too small an increment gives very fine coverage of the parameter space, too large a value results in very sparse coverage. The accumulator space, `acc` (initially zero), is incremented only for points whose coordinates lie within the specified range (in this case the center cannot lie outside the original image).

```
%Hough Transform for Circles

function HTCircle(inputimage,r)

%image size
[rows,columns]=size(inputimage);

%accumulator
acc=zeros(rows,columns);

%image
for x=1:columns
  for y=1:rows
    if(inputimage(y,x)==0)
      for ang=0:360
        t=(ang*pi)/180;
        x0=round(x-r*cos(t));
        y0=round(y-r*sin(t));
        if(x0<columns & x0>0 & y0<rows & y0>0)
          acc(y0,x0)=acc(y0,x0)+1;
        end
      end
    end
  end
end
```

CODE 5.5

Implementation of the HT for circles.

The application of the HT for circles is illustrated in Figure 5.20. Figure 5.20(a) shows an image with a synthetic circle. In this figure, the edges are complete and well defined. The result of the HT process is shown in Figure 5.20(d). The peak of the accumulator space is at the center of the circle. Note that votes exist away from

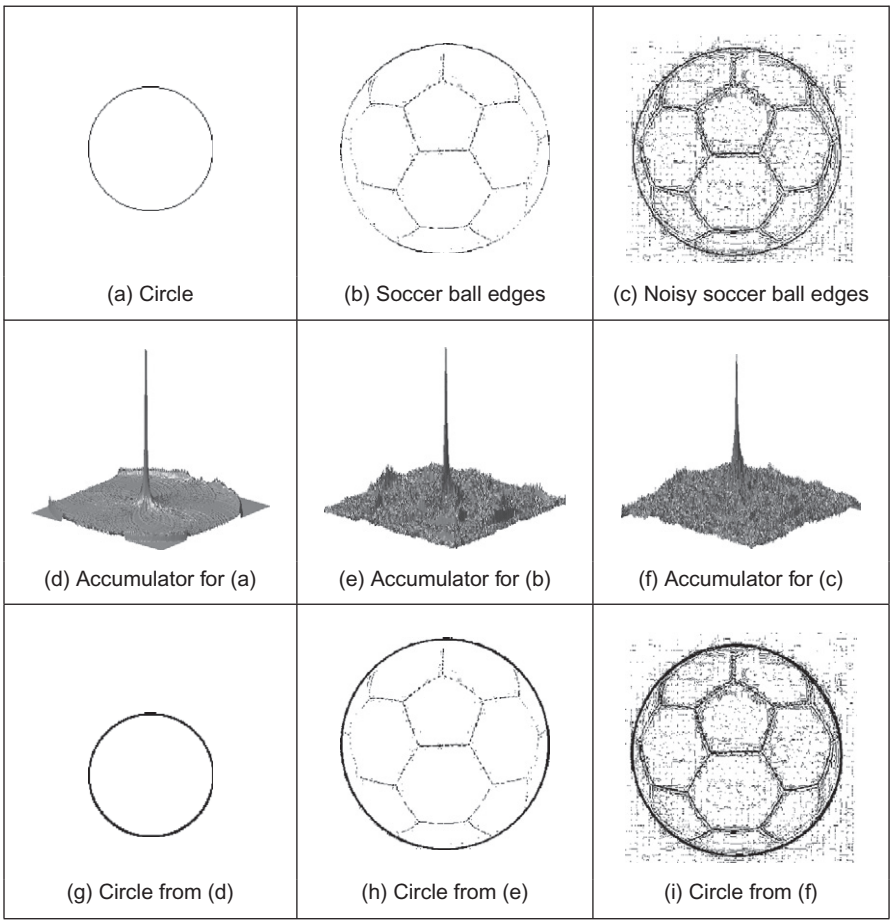


FIGURE 5.20

Applying the HT for circles.

the circle's center and rise toward the locus of the actual circle, though these background votes are much less than the actual peak. Figure 5.20(b) shows an example of data containing occlusion and noise. The image in Figure 5.20(c) corresponds to the same scene, but the noise level has been increased by changing the threshold value in the edge detection process. The accumulators for these two images are shown in Figure 5.20(e) and (f) and the circles related to the parameter space peaks are superimposed (in black) on the edge images in Figure 5.20(g)−(i). We can see that the HT has the ability to tolerate occlusion and noise. In Figure 5.20(c), there are many edge points which imply that the

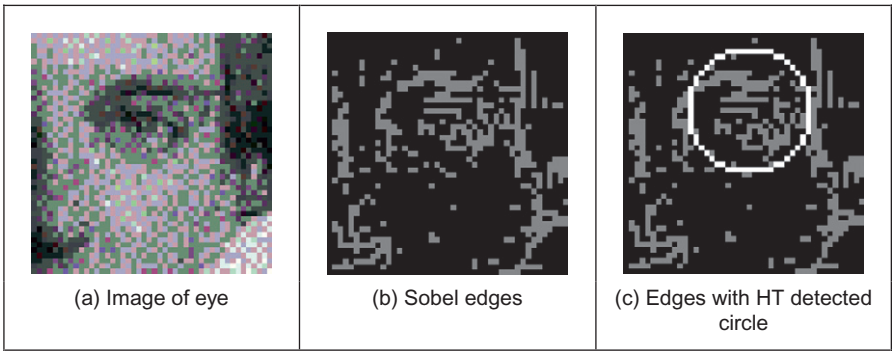


FIGURE 5.21

Using the HT for circles.

amount of processing time increases. The HT will detect the circle (provide the right result) as long as more points are in a circular locus described by the parameters of the target circle than there are on any other circle. This is exactly the same performance as for the HT for lines, as expected, and is consistent with the result of template matching.

In application code, *Bresenham's algorithm* for discrete circles (Bresenham, 1977) can be used to draw the circle of votes, rather than use the polar implementation of Eq. (5.32). This ensures that the complete locus of points is drawn and avoids need to choose a value for increase in the angle used to trace the circle. Bresenham's algorithm can be used to generate the points in one octant, since the remaining points can be obtained by reflection. Again, backmapping can be used to determine which points contributed to the extracted circle.

An additional example of the circle HT extraction is shown in Figure 5.21. Figure 5.21(a) is again a real image (albeit, one with low resolution) which was processed by Sobel edge detection and thresholded to give the points in Figure 5.21(b). The circle detected by application of `HTCircle` with radius 5 pixels is shown in Figure 5.21(c) superimposed on the edge data. The extracted circle can be seen to **match** the edge data well. This highlights the two major advantages of the HT (and of template matching): its ability to handle **noise** and **occlusion**. Note that the HT merely finds the circle with the maximum number of points; it is possible to include other constraints to control the circle selection process, such as gradient direction for objects with known illumination profile. In the case of the human eye, the (circular) iris is usually darker than its white surroundings.

Figure 5.21 also shows some of the difficulties with the HT, namely that it is essentially an implementation of template matching, and does not use some of the

richer stock of information available in an image. For example, we might know constraints on **size**; the largest size and iris would be in an image like Figure 5.21. Also, we know some of the **topology**: the eye region contains two ellipsoidal structures with a circle in the middle. We might also know **brightness** information: the pupil is darker than the surrounding iris. These factors can be formulated as **constraints** on whether edge points can vote within the accumulator array. A simple modification is to make the votes proportional to edge magnitude, in this manner, points with high contrast will generate more votes and hence have more significance in the voting process. In this way, the feature extracted by the HT can be arranged to suit a particular application.

5.5.4 HT for ellipses

Circles are very important in shape detection since many objects have a circular shape. However, because of the camera's viewpoint, circles do not always look like circles in images. Images are formed by mapping a shape in 3D space into a plane (the image plane). This mapping performs a perspective transformation. In this process, a circle is deformed to look like an ellipse. We can define the mapping between the circle and an ellipse by a similarity transformation. That is,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\rho) & \sin(\rho) \\ -\sin(\rho) & \cos(\rho) \end{bmatrix} \begin{bmatrix} S_x \\ S_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{5.33}$$

where (x',y') define the coordinates of the circle in Eq. (5.31), ρ represents the orientation, (S_x,S_y) a scale factor and (t_x,t_y) a translation. If we define

$$\begin{array}{lll} a_0 = t_x & a_x = S_x \cos(\rho) & b_x = S_y \sin(\rho) \\ b_0 = t_y & a_y = -S_x \sin(\rho) & b_y = S_y \cos(\rho) \end{array} \tag{5.34}$$

then the circle is deformed into

$$\begin{array}{l} x = a_0 + a_x \cos(\theta) + b_x \sin(\theta) \\ y = b_0 + a_y \cos(\theta) + b_y \sin(\theta) \end{array} \tag{5.35}$$

This equation corresponds to the polar representation of an ellipse. This polar form contains six parameters $(a_0,b_0,a_x,b_x,a_y,b_y)$ that characterize the shape of the ellipse. θ is not a free parameter and it only addresses a particular point in the locus of the ellipse (just as it was used to trace the circle in Eq. (5.32)). However, one parameter is redundant since it can be computed by considering the orthogonality (independence) of the axes of the ellipse (the product $a_xb_x + a_yb_y = 0$ which is one of the known properties of an ellipse). Thus, an ellipse is defined by its center (a_0,b_0) and three of the axis parameters (a_x,b_x,a_y,b_y). This gives five

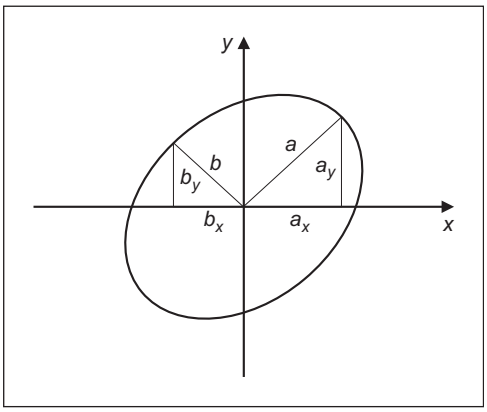


FIGURE 5.22

Definition of ellipse axes.

parameters which is intuitively correct since an ellipse is defined by its center (2 parameters), it size along both axes (2 more parameters) and its rotation (1 parameter). In total this states that 5 parameters describe an ellipse, so our three axis parameters must jointly describe size and rotation. In fact, the axis parameters can be related to the orientation and the length along the axes by

$$\tan(\rho) = \frac{a_y}{a_x} \quad a = \sqrt{a_x^2 + a_y^2} \quad b = \sqrt{b_x^2 + b_y^2} \tag{5.36}$$

where (a,b) are the axes of the ellipse, as illustrated in Figure 5.22.

In a similar way to Eq. (5.31), Eq. (5.35) can be used to generate the mapping function in the HT. In this case, the location of the center of the ellipse is given by

$$\begin{aligned} a_0 &= x - a_x \cos(\theta) + b_x \sin(\theta) \\ b_0 &= y - a_y \cos(\theta) + b_y \sin(\theta) \end{aligned} \tag{5.37}$$

The location is dependent on three parameters, thus the mapping defines the trace of a hypersurface in a 5D space. This space can be very large. For example, if there are 100 possible values for each of the five parameters, the 5D accumulator space contains 10^{10} values. This is 10 GB of storage, which is of course tiny nowadays (at least, when someone else pays!). Accordingly, there has been much interest in ellipse detection techniques which use much less space and operate much faster than direct implementation of Eq. (5.37).

Code 5.6 shows the implementation of the HT mapping for ellipses. The function `HTEllipse` computes the center parameters for an ellipse without rotation and

with fixed axis length given as arguments. Thus, the implementation uses a 2D accumulator. In practice, in order to locate an ellipse, it is necessary to try all

```
%Hough Transform for Ellipses

function HTEllipse(inputimage,a,b)

%image size
[rows,columns]=size(inputimage);

%accumulator
acc=zeros(rows,columns);

%image
for x=1:columns
  for y=1:rows
    if(inputimage(y,x)==0)
      for ang=0:360
        t=(ang*pi)/180;
        x0=round(x-a*cos(t));
        y0=round(y-b*sin(t));
        if(x0<columns & x0>0 & y0<rows & y0>0)
          acc(y0,x0)=acc(y0,x0)+1;
        end
      end
    end
  end
end
```

CODE 5.6

Implementation of the HT for ellipses.

potential values of axis length. This is computationally impossible unless we limit the computation to a few values.

Figure 5.23 shows three examples of the application of the ellipse extraction process described in Code 5.6. The first example (Figure 5.23(a)) illustrates the case of a perfect ellipse in a synthetic image. The array in Figure 5.23(d) shows a prominent peak whose position corresponds to the center of the ellipse. The examples in Figure 5.23(b) and (c) illustrate the use of the HT to locate a circular form when the image has an oblique view. Each example was obtained by using a

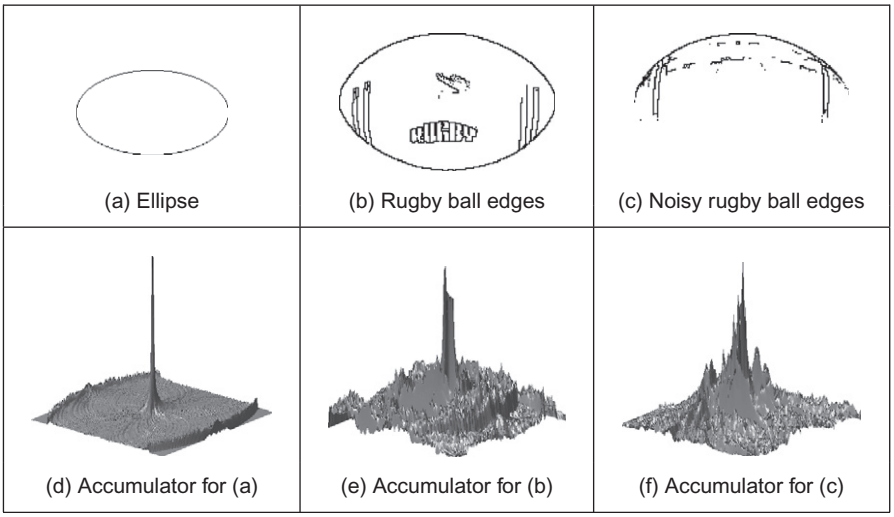


FIGURE 5.23

Applying the HT for ellipses.

different threshold in the edge detection process. Figure 5.23(c) contains more noise data that in turn gives rise to more noise in the accumulator. We can observe that there is more than one ellipse to be located in these two figures. This gives rise to the other high values in the accumulator space. As with the earlier examples for line and circle extraction, there is again scope for interpreting the accumulator space, to discover which structures produced particular parameter combinations.

5.5.5 Parameter space decomposition

The HT gives the same (optimal) result as template matching and even though it is faster, it still requires significant computational resources. In the previous sections, we saw that as we increase the complexity of the curve under detection, the computational requirements increase in an exponential way. Thus, the HT becomes less practical. For this reason, most of the research in the HT has focused on the development of techniques aimed to reduce its computational complexity (Illingworth and Kittler, 1988; Leavers, 1993). One important way to reduce the computation has been the use of geometric properties of shapes to decompose the parameter space. Several techniques have used different geometric properties.

These geometric properties are generally defined by the relationship between points and derivatives.

5.5.5.1 Parameter space reduction for lines

For a line, the accumulator space can be reduced from 2D to 1D by considering that we can compute the slope from the information of the image. The slope can be computed either by using the **gradient direction** at a point or by considering a pair of points. That is,

$$m = \varphi \quad \text{or} \quad m = \frac{y_2 - y_1}{x_2 - x_1} \tag{5.38}$$

where φ is the gradient direction at the point. In the case of two points, by considering Eq. (5.24), we have

$$c = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \tag{5.39}$$

Thus, according to Eq. (5.29), one of the parameters of the polar representation for lines, θ, is now given by

$$\theta = -\tan^{-1}\left[\frac{1}{\varphi}\right] \quad \text{or} \quad \theta = \tan^{-1}\left[\frac{x_1 - x_2}{y_2 - y_1}\right] \tag{5.40}$$

These equations do not depend on the other parameter ρ and they provide alternative mappings to gather evidence. That is, they decompose the parametric space, such that the two parameters θ and ρ are now **independent**. The use of edge direction information constitutes the base of the line extraction method presented by O'Gorman and Clowes (1976). The use of pairs of points can be related to the definition of the randomized HT (Xu et al., 1990). Obviously, the number of feature points considered corresponds to all the combinations of points that form pairs. By using statistical techniques, it is possible to reduce the space of points in order to consider a representative sample of the elements. That is, a subset which provides enough information to obtain the parameters with predefined and small estimation errors.

Code 5.7 shows the implementation of the parameter space decomposition for the HT for lines. The slope of the line is computed by considering a pair of points. Pairs of points are restricted to a neighborhood of 5 by 5 pixels. The implementation of Eq. (5.40) gives values between -90° and 90°. Since our accumulators only can store positive values, then we add 90° to all values. In order to compute ρ, we use Eq. (5.28) given the value of θ computed by Eq. (5.40).

```
%Parameter Decomposition for the Hough Transform for Lines

function HTDLine(inputimage)

%image size
[rows,columns]=size(inputimage);

%accumulator
rmax=round(sqrt(rows^2+columns^2));
accro=zeros(rmax,1);
acct=zeros(180,1);

%image
for x=1:columns
  for y=1:rows
    if(inputimage(y,x)==0)
       for Nx=x-2:x+2
         for Ny=y-2:y+2
           if(x~=Nx | y~=Ny)
              if(Nx>0 & Ny>0 & Nx<columns & Ny<rows)
                 if(inputimage(Ny,Nx)==0)
                    if(Ny-y~=0)
                        t=atan((x-Nx)/(Ny-y)); %Equation (5.40)
                     else t=pi/2;
                    end
                    r=round(x*cos(t)+y*sin(t)); %Equation (5.28)

                    t=round((t+pi/2)*180/pi);
                    acct(t)=acct(t)+1;

                    if(r<rmax & r>0)
                      accro(r)=accro(r)+1;
                    end
                 end
              end
           end
         end
       end
    end
  end
end
```

CODE 5.7

Implementation of the parameter space reduction for the HT for lines.

Figure 5.24 shows the accumulators for the two parameters θ and ρ as obtained by the implementation of Code 5.7 for the images in Figure 5.15(a) and (b). The accumulators are now 1D as shown in Figure 5.24(a) and show a clear peak. The peak in the first accumulator is close to 135°. Thus, by subtracting the 90° introduced to make all values positive, we find that the slope of the line $\theta = -45°$.

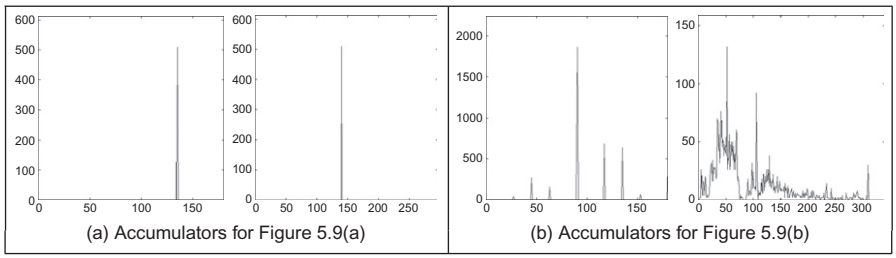


FIGURE 5.24

Parameter space reduction for the HT for lines.

The peaks in the accumulators in Figure 5.24(b) define two lines with similar slopes. The peak in the first accumulator represents the value of θ, while the two peaks in the second accumulator represent the location of the two lines. In general, when implementing parameter space decomposition, it is necessary to follow a two-step process. First, it is necessary to gather data in one accumulator and search for the maximum. Secondly, the location of the maximum value is used as parameter value to gather data of the remaining accumulator.

5.5.5.2 Parameter space reduction for circles

In the case of lines, the relationship between local information computed from an image and the inclusion of a group of points (pairs) is in an alternative analytical description which can readily be established. For more complex primitives, it is possible to include several geometric relationships. These relationships are not defined for an arbitrary set of points but include angular constraints that define relative positions between them. In general, we can consider different geometric properties of the circle to decompose the parameter space. This has motivated the development of many methods of parameter space decomposition (Aguado et al., 1996). An important geometric relationship is given by the geometry of the second directional derivatives. This relationship can be obtained by considering that Eq. (5.31) defines a position vector function. That is,

$$v(\theta) = x(\theta)\begin{bmatrix}1\\0\end{bmatrix} + y(\theta)\begin{bmatrix}0\\1\end{bmatrix} \tag{5.41}$$

where

$$x(\theta) = x_0 + r\cos(\theta); \quad y(\theta) = y_0 + r\sin(\theta) \tag{5.42}$$

In this definition, we have included the parameter of the curve as an argument in order to highlight the fact that the function defines a vector for each value of θ.

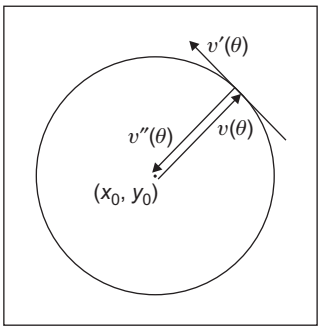


FIGURE 5.25

Definition of the first and second directional derivatives for a circle.

The end points of all the vectors trace a circle. The derivatives of Eq. (5.41) with respect to θ define the first and second directional derivatives. That is,

$$v'(\theta) = x'(\theta)\begin{bmatrix}1\\0\end{bmatrix} + y'(\theta)\begin{bmatrix}0\\1\end{bmatrix}$$

$$v''(\theta) = x''(\theta)\begin{bmatrix}1\\0\end{bmatrix} + y''(\theta)\begin{bmatrix}0\\1\end{bmatrix} \tag{5.43}$$

where

$$\begin{aligned} x'(\theta) &= -r\sin(\theta); \quad y'(\theta) = r\cos(\theta) \\ x''(\theta) &= -r\cos(\theta); \quad y''(\theta) = -r\sin(\theta) \end{aligned} \tag{5.44}$$

Figure 5.25 illustrates the definition of the first and second directional derivatives. The first derivative defines a tangential vector, while the second one is similar to the vector function, but it has reverse direction. In fact, the edge direction measured for circles can be arranged so as to point toward the center was actually the basis of one of the early approaches to reducing the computational load of the HT for circles (Kimme et al., 1975).

According to Eqs (5.42) and (5.44), we observe that the tangent of the angle of the first directional derivative denoted as $\phi'(\theta)$ is given by

$$\phi'(\theta) = \frac{y'(\theta)}{x'(\theta)} = -\frac{1}{\tan(\theta)} \tag{5.45}$$

Angles will be denoted by using the symbol ^. That is,

$$\hat{\phi}'(\theta) = \tan^{-1}(\phi'(\theta)) \tag{5.46}$$

Similarly, for the tangent of the second directional derivative, we have

$$\phi''(\theta) = \frac{y''(\theta)}{x''(\theta)} = \tan(\theta) \quad \text{and} \quad \hat{\phi}''(\theta) = \tan^{-1}(\phi''(\theta)) \tag{5.47}$$

By observing the definition of $\phi''(\theta)$, we have

$$\phi''(\theta) = \frac{y''(\theta)}{x''(\theta)} = \frac{y(\theta) - y_0}{x(\theta) - x_0} \tag{5.48}$$

This equation defines a straight line passing through the points $(x(\theta),y(\theta))$ and (x_0,y_0) and it is perhaps the most important relation in parameter space decomposition. The definition of the line is more evident by rearranging terms. That is,

$$y(\theta) = \phi''(\theta)(x(\theta) - x_0) + y_0 \tag{5.49}$$

This equation is independent of the radius parameter. Thus, it can be used to gather evidence of the location of the shape in a 2D accumulator. The HT mapping is defined by the dual form given by

$$y_0 = \phi''(\theta)(x_0 - x(\theta)) + y(\theta) \tag{5.50}$$

That is, given an image point $(x(\theta),y(\theta))$ and the value of $\phi''(\theta)$, we can generate a line of votes in the 2D accumulator (x_0,y_0). Once the center of the circle is known, then a 1D accumulator can be used to locate the radius. The key aspect of the parameter space decomposition is the method used to obtain the value of $\phi''(\theta)$ from image data. We will consider two alternative ways. First, we will show that $\phi''(\theta)$ can be obtained by edge direction information. Secondly, how it can be obtained from the information of a pair of points.

In order to obtain $\phi''(\theta)$, we can use the definition in Eqs (5.45) and (5.47). According to these equations, the tangents $\phi''(\theta)$ and $\phi'(\theta)$ are perpendicular. Thus,

$$\phi''(\theta) = -\frac{1}{\phi'(\theta)} \tag{5.51}$$

Thus, the HT mapping in Eq. (5.50) can be written in terms of gradient direction $\phi'(\theta)$ as

$$y_0 = y(\theta) + \frac{x(\theta) - x_0}{\phi'(\theta)} \tag{5.52}$$

This equation has a simple geometric interpretation illustrated in Figure 5.26(a). We can see that the line of votes passes through the points $(x(\theta),y(\theta))$ and (x_0,y_0). The slope of the line is perpendicular to the direction of gradient direction.

An alternative decomposition can be obtained by considering the geometry shown in Figure 5.26(b). In the figure we can see that if we take a pair of points (x_1,y_1) and (x_2,y_2), where $x_i = x(\theta_i)$, then the line that passes through the points has the same slope as the line at a point $(x(\theta),y(\theta))$. Accordingly,

$$\phi'(\theta) = \frac{y_2 - y_1}{x_2 - x_1} \tag{5.53}$$

where

$$\theta = \tfrac{1}{2}(\theta_1 + \theta_2) \tag{5.54}$$

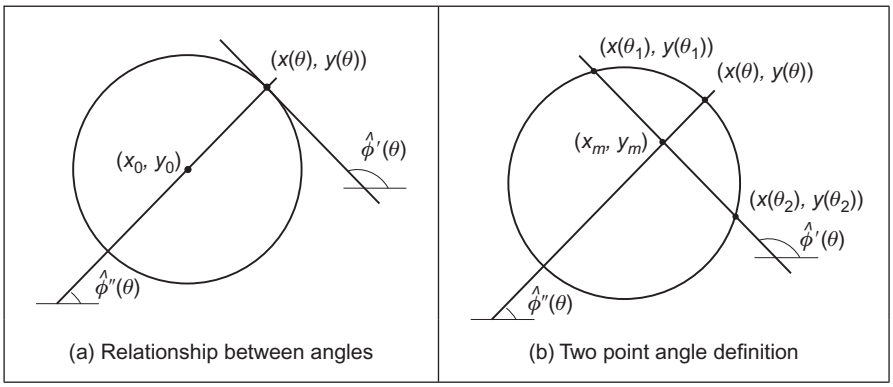


FIGURE 5.26

Geometry of the angle of the first and second directional derivatives.

Based on Eq. (5.53), we have

$$\phi''(\theta) = -\frac{x_2 - x_1}{y_2 - y_1} \tag{5.55}$$

The problem with using a pair of points is that by Eq. (5.54), we cannot know the location of the point $(x(\theta),y(\theta))$. Fortunately, the voting line also passes through the midpoint of the line between the two selected points. Let us define this point as

$$x_m = \tfrac{1}{2}(x_1 + x_2); \quad y_m = \tfrac{1}{2}(y_1 + y_2) \tag{5.56}$$

Thus, by substitution of Eq. (5.53) in Eq. (5.52) and by replacing the point $(x(\theta),y(\theta))$ by (x_m,y_m), the HT mapping can be expressed as

$$y_0 = y_m + \frac{(x_m - x_0)(x_2 - x_1)}{(y_2 - y_1)} \tag{5.57}$$

This equation does not use gradient direction information, but it is based on pairs of points. This is analogous to the parameter space decomposition of the line presented in Eq. (5.40). In that case, the slope can be computed by using gradient direction or, alternatively, by taking a pair of points. In the case of the circle, the tangent (and therefore the angle of the second directional derivative) can be computed by the gradient direction (i.e., Eq. (5.51)) or by a pair of points (i.e., Eq. (5.55)). However, it is important to note that there are some other combinations of parameter space decomposition (Aguado, 1996).

Code 5.8 shows the implementation of the parameter space decomposition for the HT for circles. The implementation only detects the position of the circle and it gathers evidence by using the mapping in Eq. (5.57). Pairs of points are

```
%Parameter Decomposition for the Hough Transform for Circles

function HTDCircle(inputimage)

%image size
[rows,columns]=size(inputimage);

%accumulator
acc=zeros(rows,columns);

%gather evidence
for x1=1:columns
  for y1=1:rows
    if(inputimage(y1,x1)==0)
       for x2=x1-12:x1+12
         for y2=y1-12:y1+12
           if(abs(x2-x1)>10 | abs(y2-y1)>10)
             if(x2>0 & y2>0 & x2<columns & y2<rows)
               if(inputimage(y2,x2)==0)
                  xm=(x1+x2)/2;   ym=(y1+y2)/2;
                   if(y2-y1~=0)   m=((x2-x1)/(y2-y1));
                      else   m=99999999;
                   end

                  if(m>-1 & m<1)
                    for x0=1:columns
                        y0=round(ym+m*(xm-x0));
                        if(y0>0 & y0<rows)
                           acc(y0,x0)=acc(y0,x0)+1;
                        end
                    end
                  else
                    for y0=1:rows
                        x0= round(xm+(ym-y0)/m);
                        if(x0>0 & x0<columns)
                           acc(y0,x0)=acc(y0,x0)+1;
                        end
                    end
                  end
               end
             end
           end
         end
       end
    end
  end
end
```

CODE 5.8

Parameter space reduction for the HT for circles.

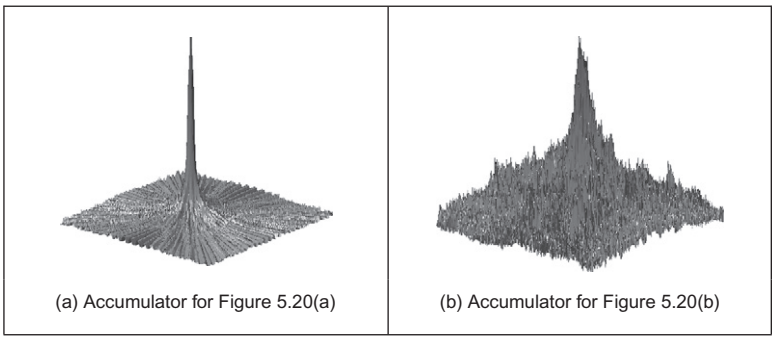


FIGURE 5.27

Parameter space reduction for the HT for circles.

restricted to a neighborhood between 10×10 pixels and 12×12 pixels. We avoid using pixels that are close to each other since they do not produce accurate votes. We also avoid using pixels that are far away from each other, since by distance it is probable that they do not belong to the same circle and would only increase the noise in the accumulator. In order to trace the line, we use two equations that are selected according to the slope.

Figure 5.27 shows the accumulators obtained by the implementation of Code 5.8 for the images in Figure 5.20(a) and (b). Both accumulators show a clear peak that represents the location of the circle. Small peaks in the background of the accumulator in Figure 5.27(b) correspond to circles with only a few points. In general, there is a compromise between the **width** of the peak and the **noise** in the accumulator. The peak can be made narrower by considering pairs of points that are more widely spaced. However, this can also increases the level of background noise. Background noise can be reduced by taking points that are closer together, but this makes the peak wider.

5.5.5.3 Parameter space reduction for ellipses

Part of the simplicity in the parameter decomposition for circles comes from the fact that circles are (naturally) isotropic. Ellipses have more free parameters and are geometrically more complex. Thus, geometrical properties involve more complex relationships between points, tangents, and angles. However, they maintain the geometric relationship defined by the angle of the second derivative. According to Eqs (5.41) and (5.43), the vector position and directional derivatives of an ellipse in Eq. (5.35) have the components

$$\begin{aligned} x'(\theta) &= -a_x \sin(\theta) + b_x \cos(\theta); \quad y'(\theta) = -a_y \sin(\theta) + b_y \cos(\theta) \\ x''(\theta) &= -a_x \cos(\theta) - b_x \sin(\theta); \quad y''(\theta) = -a_y \cos(\theta) - b_y \sin(\theta) \end{aligned} \tag{5.58}$$

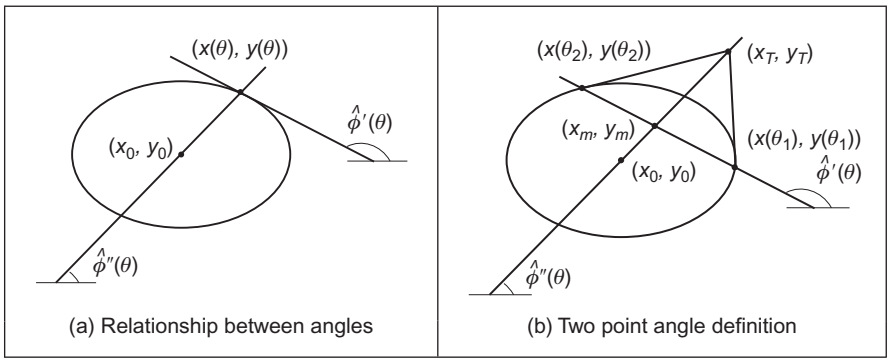


FIGURE 5.28

Geometry of the angle of the first and second directional derivatives.

The tangent angles of the first and second directional derivatives are given by

$$\begin{aligned}\phi'(\theta) &= \frac{y'(\theta)}{x'(\theta)} = \frac{-a_y \cos(\theta) + b_y \sin(\theta)}{-a_x \cos(\theta) + b_x \sin(\theta)} \\ \phi''(\theta) &= \frac{y''(\theta)}{x''(\theta)} = \frac{-a_y \cos(\theta) - b_y \sin(\theta)}{-a_x \cos(\theta) - b_x \sin(\theta)}\end{aligned} \tag{5.59}$$

By considering Eq. (5.58), we have that Eq. (5.48) is also valid for an ellipse. That is,

$$\frac{y(\theta) - y_0}{x(\theta) - x_0} = \phi''(\theta) \tag{5.60}$$

The geometry of the definition in this equation is illustrated in Figure 5.28(a). As in the case of circles, this equation defines a line that passes through the points $(x(\theta),y(\theta))$ and (x_0,y_0). However, in the case of the ellipse, the angles $\hat{\phi}'(\theta)$ and $\hat{\phi}''(\theta)$ are not orthogonal. This makes the computation of $\hat{\phi}''(\theta)$ more complex. In order to obtain $\phi''(\theta)$, we can extend the geometry presented in Figure 5.26(b). That is, we take a pair of points to define a line whose slope defines the value of $\phi'(\theta)$ at another point. This is illustrated in Figure 5.28(b). The line in Eq. (5.60) passes through the middle point (x_m,y_m). However, it is not orthogonal to the tangent line. In order to obtain an expression of the HT mapping, we will first show that the relationship in Eq. (5.54) is also valid for ellipses. Then we will use this equation to obtain $\phi''(\theta)$.

The relationships in Figure 5.28(b) do not depend on the orientation or position of the ellipse. Thus, the three points can be defined by

$$\begin{aligned} x_1 &= a_x \cos(\theta_1); \quad x_2 = a_x \cos(\theta_2); \quad x(\theta) = a_x \cos(\theta) \\ y_1 &= b_x \sin(\theta_1); \quad y_2 = b_x \sin(\theta_2); \quad y(\theta) = b_x \sin(\theta) \end{aligned} \tag{5.61}$$

The point $(x(\theta),y(\theta))$ is given by the intersection of the line in Eq. (5.60) with the ellipse. That is,

$$\frac{y(\theta)-y_0}{x(\theta)-x_0}=\frac{a_x}{b_y}\frac{y_m}{x_m} \tag{5.62}$$

By substitution of the values of (x_m,y_m) defined as the average of the coordinates of the points (x_1,y_1) and (x_2,y_2) in Eq. (5.56), we have

$$\tan(\theta)=\frac{a_x}{b_y}\frac{b_y\sin(\theta_1)+b_y\sin(\theta_2)}{a_x\cos(\theta_1)+a_x\cos(\theta_2)} \tag{5.63}$$

Thus,

$$\tan(\theta)=\tan(\tfrac{1}{2}(\theta_1+\theta_2)) \tag{5.64}$$

From this equation it is evident that the relationship in Eq. (5.54) is also valid for ellipses. Based on this result, the tangent angle of the second directional derivative can be defined as

$$\phi''(\theta)=\frac{b_y}{a_x}\tan(\theta) \tag{5.65}$$

By substitution in Eq. (5.62), we have

$$\phi''(\theta)=\frac{y_m}{x_m} \tag{5.66}$$

This equation is valid when the ellipse is not translated. If the ellipse is translated, then the tangent of the angle can be written in terms of the points (x_m,y_m) and (x_T,y_T) as

$$\phi''(\theta)=\frac{y_T-y_m}{x_T-x_m} \tag{5.67}$$

By considering that the point (x_T,y_T) is the intersection point of the tangent lines at (x_1,y_1) and (x_2,y_2), we obtain

$$\phi''(\theta)=\frac{AC+2BD}{2A+BC} \tag{5.68}$$

where

$$\begin{aligned} A&=y_1-y_2; \quad B=x_1-x_2\\ C&=\phi_1+\phi_2; \quad D=\phi_1\cdot\phi_2 \end{aligned} \tag{5.69}$$

and ϕ_1, ϕ_2 are the slopes of the tangent line to the points. Finally, by considering Eq. (5.60), the HT mapping for the center parameter is defined as

$$y_0=y_m+\frac{AC+2BD}{2A+BC}(x_0-x_m) \tag{5.70}$$

This equation can be used to gather evidence that is independent of rotation or scale. Once the location is known, a 3D parameter space is necessary to obtain

the remaining parameters. However, these parameters can also be computed independently using two 2D parameter spaces (Aguado et al., 1996). Of course you can avoid using the gradient direction in Eq. (5.68) by including more points. In fact, the tangent $\phi''(\theta)$ can be computed by taking four points (Aguado, 1996). However, the inclusion of more points generally leads to more background noise in the accumulator.

Code 5.9 shows the implementation of the ellipse location mapping in Eq. (5.57). As in the case of the circle, pairs of points need to be restricted to a

```
%Parameter Decomposition for Ellipses
function HTDEllipse(inputimage)

%image size
[rows,columns]=size(inputimage);

%edges
[M,Ang]=Edges(inputimage);
M=MaxSupr(M,Ang);

%accumulator
acc=zeros(rows,columns);

%gather evidence
for x1=1:columns
  for y1=1:1:rows
    if(M(y1,x1)~=0)
      for i=0:60
        x2=x1-i; y2=y1-i;
        incx=1; incy=0;
        for k=0: 8*i-1
          if(x2>0 & y2>0 & x2<columns & y2<rows)
          if M(y2,x2)~=0

            m1=Ang(y1,x1); m2=Ang(y2,x2);

            if(abs(m1-m2)>.2)

              xm=(x1+x2)/2; ym=(y1+y2)/2;
              m1=tan(m1); m2=tan(m2);

              A=y1-y2; B=x1-x2;
              C=m1+m2; D=m1*m2;
              N=(2*A+B*C);
              if N~=0
```

CODE 5.9

Implementation of the parameter space reduction for the HT for ellipses.

```
                m=(A*C+2*B*D)/N;
              else
                m=99999999;
              end;

            if(m>-1 & m<1)
              for x0=1:columns
                y0=round(ym+m*(xm-x0));
                if(y0>0 & y0<rows)
                  acc(y0,x0)=acc(y0,x0)+1;
                end
              end
            else
              for y0=1:rows
                x0= round(xm+(ym-y0)/m);
                if(x0>0 & x0<columns)
                  acc(y0,x0)=acc(y0,x0)+1;
                end
              end
            end % if abs
          end  % if M
          end

          x2=x2+incx; y2=y2+incy;

          if x2>x1+i
            x2=x1+i;
            incx=0;     incy=1;
            y2=y2+incy;
          end

          if y2>y1+i
            y2=y1+i;
            incx=-1;    incy=0;
            x2=x2+incx;
          end

          if x2<x1-i
            x2=x1-i;
            incx=0;     incy=-1;
            y2=y2+incy;
          end
        end % for k
      end % for i
    end % if (x1,y1)
  end % y1
end %x1
```

CODE 5.9

(Continued)

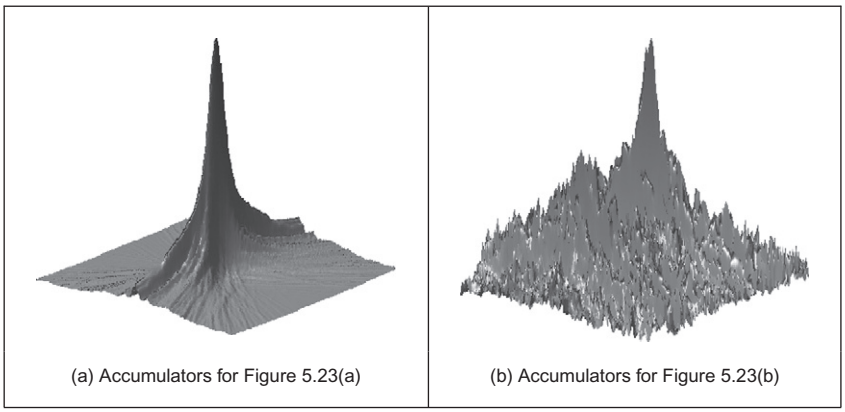


FIGURE 5.29

Parameter space reduction for the HT for ellipses.

neighborhood. In the implementation, we consider pairs at a fixed distance given by the variable *i*. Since we are including gradient direction information, the resulting peak is generally quite wide. Again, the selection of the distance between points is a compromise between the level of background noise and the width of the peak.

Figure 5.29 shows the accumulators obtained by the implementation of Code 5.9 for the images in Figure 5.23(a) and (b). The peak represents the location of the ellipses. In general, there is noise and the accumulator is wide. This is for two main reasons. First, when the gradient direction is not accurate, then the line of votes does not pass exactly over the center of the ellipse. This forces the peak to become wider with less height. Secondly, in order to avoid numerical instabilities, we need to select points that are well separated. However, this increases the probability that the points do not belong to the same ellipse, thus generating background noise in the accumulator.

5.5.6 Generalized HT

Many shapes are far more complex than lines, circles, or ellipses. It is often possible to partition a complex shape into several geometric primitives, but this can lead to a highly complex data structure. In general it is more convenient to extract the whole shape. This has motivated the development of techniques that can find **arbitrary** shapes using the evidence-gathering procedure of the HT. These techniques again give results equivalent to those delivered by matched template filtering, but with the computational advantage of the evidence-gathering approach. An early approach offered only limited capability only for arbitrary shapes (Merlin and Farber, 1975). The full mapping is called the *generalized HT* (GHT) (Ballard, 1981) and can be used to locate arbitrary shapes with unknown **position**,

size, and **orientation**. The GHT can be formally defined by considering the duality of a curve. One possible implementation can be based on the discrete representation given by tabular functions. These two aspects are explained in the following two sections.

5.5.6.1 Formal definition of the GHT

The formal analysis of the HT provides the route for generalizing it to arbitrary shapes. We can start by generalizing the definitions in Eq. (5.41). In this way a model shape can be defined by a curve:

$$v(\theta) = x(\theta)\begin{bmatrix}1\\0\end{bmatrix} + y(\theta)\begin{bmatrix}0\\1\end{bmatrix} \tag{5.71}$$

For a circle, for example, we have $x(\theta) = r\cos(\theta)$ and $y(\theta) = r\sin(\theta)$. Any shape can be represented by following a more complex definition of $x(\theta)$ and $y(\theta)$.

In general, we are interested in matching the model shape against a shape in an image. However, the shape in the image has a different location, orientation, and scale. Originally the GHT defines a scale parameter in the x and y directions, but due to computational complexity and practical relevance, the use of a single scale has become much more popular. Analogous to Eq. (5.33), we can define the image shape by considering translation, rotation, and change of scale. Thus, the shape in the image can be defined as

$$\omega(\theta, b, \lambda, \rho) = b + \lambda\mathbf{R}(\rho)v(\theta) \tag{5.72}$$

where $b = (x_0,y_0)$ is the translation vector, λ is a scale factor, and $\mathbf{R}(\rho)$ is a rotation matrix (as in Eq. (5.31)). Here we have included explicitly the parameters of the transformation as arguments, but to simplify the notation they will be omitted later. The shape of $\omega(\theta,b,\lambda,\rho)$ depends on four parameters. Two parameters define the location b, plus the rotation and scale. It is important to note that θ does not define a free parameter, but it only traces the curve.

In order to define a mapping for the HT, we can follow the approach used to obtain Eq. (5.35). Thus, the location of the shape is given by

$$b = \omega(\theta) - \lambda\mathbf{R}(\rho)v(\theta) \tag{5.73}$$

Given a shape $\omega(\theta)$ and a set of parameters b, λ, and ρ, this equation defines the location of the shape. However, we do not know the shape $\omega(\theta)$ (since it depends on the parameters that we are looking for), but we only have a point in the curve. If we call $\omega_i = (\omega_{xi},\omega_{yi})$ the point in the image, then

$$b = \omega_i - \lambda\mathbf{R}(\rho)v(\theta) \tag{5.74}$$

defines a system with four unknowns and with as many equations as points in the image. In order to find the solution, we can gather evidence by using a 4D accumulator space. For each potential value of b, λ, and ρ, we trace a point spread function by considering all the values of θ, i.e., all the points in the curve $v(\theta)$.

In the GHT, the gathering process is performed by adding an extra constraint to the system that allows us to match points in the image with points in the model shape. This constraint is based on gradient direction information and can be explained as follows. We said that ideally, we would like to use Eq. (5.73) to gather evidence. For that we need to know the shape $\omega(\theta)$ and the model $\upsilon(\theta)$, but we only know the discrete points ω_i and we have supposed that these are the same as the shape, i.e., $\omega(\theta) = \omega_i$. Based on this assumption, we then consider all the potential points in the model shape, $\upsilon(\theta)$. However, this is not necessary since we only need the point in the model,$\upsilon(\theta)$, that corresponds to the point in the shape, $\omega(\theta)$. We cannot know the point in the shape, $\upsilon(\theta)$, but we can compute some properties from the model and image. Then, we can check whether these properties are similar at the point in the model and at a point in the image. If they are indeed **similar**, the points might **correspond**: if they do we can gather evidence of the parameters of the shape. The GHT considers as feature the gradient direction at the point. We can generalize Eqs (5.45) and (5.46) to define the gradient direction at a point in the arbitrary model. Thus,

$$\phi'(\theta) = \frac{y'(\theta)}{x'(\theta)} \quad \text{and} \quad \hat{\phi}'(\theta) = \tan^{-1}(\phi'(\theta)) \tag{5.75}$$

Thus Eq. (5.73) is true only if the gradient direction at a point in the image matches the rotated gradient direction at a point in the (rotated) model, i.e.,

$$\phi_i' = \hat{\phi}'(\theta) - \rho \tag{5.76}$$

where $\hat{\phi}'(\theta)$ is the angle at the point ω_i. Note that according to this equation, gradient direction is independent of scale (in theory at least) and it changes in the same ratio as rotation. We can constrain Eq. (5.74) to consider only the points $\upsilon(\theta)$ for which

$$\phi_i' - \hat{\phi}'(\theta) + \rho = 0 \tag{5.77}$$

That is, a point spread function for a given edge point ω_i is obtained by selecting a subset of points in $\upsilon(\theta)$ such that the edge direction at the image point rotated by ρ equals the gradient direction at the model point. For each point ω_i and selected point in $\upsilon(\theta)$, the point spread function is defined by the HT mapping in Eq. (5.74).

5.5.6.2 Polar definition

Equation (5.74) defines the mapping of the HT in Cartesian form. That is, it defines the votes in the parameter space as a pair of coordinates (x,y). There is an alternative definition in polar form. The polar implementation is more common than the Cartesian form (Hecker and Bolle, 1994; Sonka et al., 1994). The advantage of the polar form is that it is easy to implement since changes in rotation and scale correspond to addition in the angle−magnitude representation. However, ensuring that the polar vector has the correct direction incurs more complexity.

Equation (5.74) can be written in a form that combines rotation and scale as

$$b = \omega(\theta) - \gamma(\lambda, \rho) \tag{5.78}$$

where $\gamma^{\mathrm{T}}(\lambda,\rho) = [\gamma_x(\lambda,\rho) \quad \gamma_y(\lambda\rho)]$ and where the combined rotation and scale is

$$\begin{aligned} \gamma_x(\lambda, \rho) &= \lambda(x(\theta)\cos(\rho) - y(\theta)\sin(\rho)) \\ \gamma_y(\lambda, \rho) &= \lambda(x(\theta)\sin(\rho) + y(\theta)\cos(\rho)) \end{aligned} \tag{5.79}$$

This combination of rotation and scale defines a vector, $\gamma(\lambda,\rho)$, whose tangent angle and magnitude are given by

$$\tan(\alpha) = \frac{\gamma_y(\lambda, \rho)}{\gamma_x(\lambda, \rho)}; \quad r = \sqrt{\gamma_x^2(\lambda, \rho) + \gamma_y^2(\lambda, \rho)} \tag{5.80}$$

The main idea here is that if we know the values for α and r, then we can gather evidence by considering Eq. (5.78) in polar form. That is,

$$b = \omega(\theta) - r\, \mathrm{e}^{j\alpha} \tag{5.81}$$

Thus, we should focus on computing values for α and r. After some algebraic manipulation, we have

$$\alpha = \phi(\theta) + \rho; \quad r = \lambda\, \Gamma(\theta) \tag{5.82}$$

where

$$\phi(\theta) = \tan^{-1}\left(\frac{y(\theta)}{x(\theta)}\right); \quad \Gamma(\theta) = \sqrt{x^2(\theta) + y^2(\theta)} \tag{5.83}$$

In this definition, we must include the constraint defined in Eq. (5.77). That is, we gather evidence only when the gradient direction is the **same**. Note that the square root in the definition of the magnitude in Eq. (5.83) can have positive and negative values. The sign must be selected in a way that the vector has the correct direction.

5.5.6.3 The GHT technique

Equations (5.74) and (5.81) define an HT mapping function for arbitrary shapes. The geometry of these equations is shown in Figure 5.30. Given an image point ω_i, we have to find a displacement vector $\gamma(\lambda,\rho)$. When the vector is placed at ω_i, then its end is at the point b. In the GHT jargon, this point called the reference point. The vector $\gamma(\lambda,\rho)$ can be easily obtained as $\lambda R(\rho)\upsilon(\theta)$ or alternative as $r\mathrm{e}^{\alpha}$. However, in order to evaluate these equations, we need to know the point $\upsilon(\theta)$. This is the crucial step in the evidence-gathering process. Note the remarkable similarity between Figures 5.26(a), 5.28(a) and 5.30(a). This is not a coincidence, but Eq. (5.60) is a particular case of Eq. (5.73).

The process of determining $\upsilon(\theta)$ centers on solving Eq. (5.76). According to this equation, since we know $\hat{\phi}'_i$, we need to find the point $\upsilon(\theta)$ whose gradient direction is $\hat{\phi}'_i + \rho = 0$. Then we must use $\upsilon(\theta)$ to obtain the displacement vector

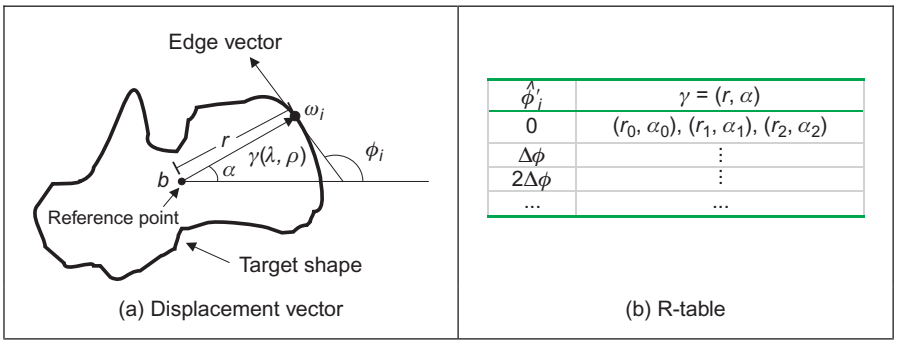


FIGURE 5.30

Geometry of the GHT.

$\gamma(\lambda,\rho)$. The GHT precomputes the solution of this problem and stores it an array called the *R-table*. The R-table stores for each value of $\hat{\phi}'_i$ the vector $\gamma(\lambda,\rho)$ for $\rho = 0$ and $\lambda = 1$. In polar form, the vectors are stored as a magnitude direction pair and in Cartesian form as a coordinate pair.

The possible range for $\hat{\phi}'_i$ is between $-\pi/2$ and $\pi/2$ radians. This range is split into N equispaced slots or bins. These slots become rows of data in the R-table. The edge direction at each border point determines the appropriate row in the R-table. The length, r, and direction, α, from the reference point is entered into a new column element, at that row, for each border point in the shape. In this manner, the N rows of the R-table have elements related to the border information, elements for which there is no information containing null vectors. The length of each row is given by the number of edge points that have the edge direction corresponding to that row; the total number of elements in the R-table equals the number of edge points above a chosen threshold. The **structure** of the R-table for N edge direction bins and m template border points is illustrated in Figure 5.30(b).

The process of **building** the R-table is illustrated in Code 5.10. In this code, we implement the Cartesian definition given in Eq. (5.74). According to this equation, the displacement vector is given by

$$\gamma(1,0) = \omega(\theta) - b \tag{5.84}$$

The matrix **T** stores the coordinates of $\gamma(1,0)$. This matrix is expanded to accommodate all the computed entries.

Code 5.11 shows the implementation of the gathering process of the GHT. In this case we use the Cartesian definition in Eq. (5.74). The coordinates of points given by evaluation of all R-table points for the particular row indexed by the gradient magnitude are used to increment cells in the accumulator array. The maximum number of votes occurs at the location of the original reference point. After all edge points have been inspected, the location of the shape is given by the maximum of an accumulator array.

```
%R-Table

function T=RTable(entries,inputimage)

%image size
[rows,columns]=size(inputimage);

%edges
[M,Ang]=Edges(inputimage);
M=MaxSupr(M,Ang);

%compute reference point
xr=0; yr=0; p=0;
for x=1:columns
  for y=1:rows
     if(M(y,x)~=0)
        xr=xr+x;
        yr=yr+y;
        p=p+1;
     end
  end
end
xr=round(xr/p);
yr=round(yr/p);

%accumulator
D=pi/entries;

s=0;            % number of entries in the table
t=[];
F=zeros(entries,1); % number of entries in the row

% for each edge point
for x=1:columns
   for y=1:rows
      if(M(y,x)~=0)

         phi=Ang(y,x);
         i=round((phi+(pi/2))/D);
         if(i==0) i=1; end;

         V=F(i)+1;

         if(V>s)
            s=s+1;
            T(:,:,s)=zeros(entries,2);
         end;

         T(i,1,V)=x-xr;
         T(i,2,V)=y-yr;
         F(i)=F(i)+1;

      end %if
   end % y
end% x
```

CODE 5.10

Implementation of the construction of the R-table.

```
%Generalised Hough Transform

function GHT(inputimage,RTable)

%image size
[rows,columns]=size(inputimage);

%table size
[rowsT,h,columnsT]=size(RTable);
D=pi/rowsT;

%edges
[M,Ang]=Edges(inputimage);
M=MaxSupr(M,Ang);

%accumulator
acc=zeros(rows,columns);

%for each edge point
for x=1:columns
  for y=1:rows
     if(M(y,x)~=0)

       phi=Ang(y,x);
       i=round((phi+(pi/2))/D);
       if(i==0) i=1; end;
       for j=1:columnsT
         if(RTable(i,1,j)==0 & RTable(i,2,j)==0)
           j=columnsT; %no more entries
         else
           a0=x-RTable(i,1,j); b0=y-RTable(i,2,j);
           if(a0>0 & a0<columns & b0>0 & b0<rows)
             acc(b0,a0)=acc(b0,a0)+1;
           end
         end
       end
     end %if
  end % y
end% x
```

CODE 5.11

Implementing the GHT.

Note that if we want to try other values for rotation and scale, then it is necessary to compute a table $\gamma(\lambda,\rho)$ for all potential values. However, this can be avoided by considering that $\gamma(\lambda,\rho)$ can be computed from $\gamma(1,0)$. That is, if we want to accumulate evidence for $\gamma(\lambda,\rho)$, then we use the entry indexed by $\hat{\phi}'_i + \rho$ and we rotate and scale the vector $\gamma(1,0)$. That is,

$$\begin{aligned} \gamma_x(\lambda,\rho) &= \lambda(\gamma_x(1,0)\cos(\rho) - \gamma_y(1,0)\sin(\rho)) \\ \gamma_y(\lambda,\rho) &= \lambda(\gamma_x(1,0)\sin(\rho) + \gamma_y(1,0)\cos(\rho)) \end{aligned} \tag{5.85}$$

In the case of the polar form, the angle and magnitude need to be defined according to Eq. (5.82).

The application of the GHT to detect an arbitrary shape with unknown translation is illustrated in Figure 5.31. We constructed an R-table from the template shown in Figure 5.3. The table contains 30 rows. The accumulator in Figure 5.31(c) was obtained by applying the GHT to the image in Figure 5.31(b). Since the table was obtained from a shape with the same scale and rotation than the primitive in the image, then the GHT produces an accumulator with a clear peak at the center of mass of the shape.

Although the example in Figure 5.31 shows that the GHT is an effective method for shape extraction, there are several inherent difficulties in its formulation (Grimson and Huttenglocher, 1990; Aguado et al., 2000b). The most evident problem is that the table does not provide an accurate representation when objects are scaled and translated. This is because the table implicitly assumes that the curve is represented in discrete form. Thus, the GHT maps a discrete form into a discrete parameter space. Additionally, the transformation of scale and rotation can induce other discretization errors. This is because when discrete images are mapped to be larger, or when they are rotated, loci which are unbroken sets of points rarely map to unbroken sets in the new image. Another important problem is the excessive computations required by the 4D parameter space. This makes the technique impractical. Also, the GHT is clearly dependent on the accuracy of

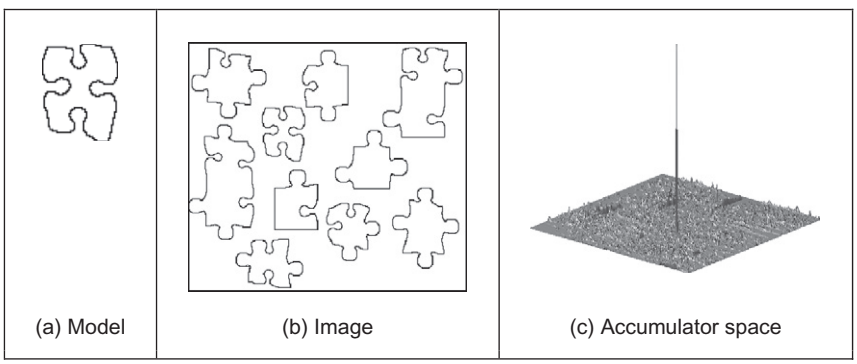


FIGURE 5.31

Example of the GHT.

directional information. By these factors, the results provided by the GHT can become less reliable. A solution is to use an analytic form instead of a table (Aguado et al., 1998). This avoids discretization errors and makes the technique more reliable. This also allows the extension to affine or other transformations. However, this technique requires solving for the point $v(\theta)$ in an analytic way increasing the computational load. A solution is to reduce the number of points by considering characteristics points defined as points of high curvature. However, this still requires the use of a 4D accumulator. An alternative to reduce this computational load is to include the concept of invariance in the GHT mapping.

5.5.6.4 Invariant GHT

The problem with the GHT (and other extensions of the HT) is that they are very general. That is, the HT gathers evidence for a single point in the image. However, a point on its own provides little information. Thus, it is necessary to consider a large parameter space to cover all the potential shapes defined by a given image point. The GHT improves evidence gathering by considering a point and its gradient direction. However, since gradient direction changes with rotation, the evidence gathering is improved in terms of noise handling, but little is done about computational complexity.

In order to reduce computational complexity of the GHT, we can consider replacing the gradient direction by another feature. That is, by a feature that is not affected by **rotation**. Let us explain this idea in more detail. The main aim of the constraint in Eq. (5.77) is to include gradient direction to reduce the number of votes in the accumulator by identifying a point $v(\theta)$. Once this point is known, then we obtain the displacement vector $\gamma(\lambda,\rho)$. However, for each value of rotation, we have a different point in $v(\theta)$. Now let us replace that constraint in Eq. (5.76) by a constraint of the form

$$Q(\omega_i) = Q(v(\theta)) \tag{5.86}$$

The function Q is said to be invariant and it computes a feature at the point. This feature can be, for example, the color of the point, or any other property that does not change in the model and image. By considering Eq. (5.86), Eq. (5.77) is redefined as

$$Q(\omega_i) - Q(v(\theta)) = 0 \tag{5.87}$$

That is, instead of searching for a point with the same gradient direction, we will search for the point with the same invariant feature. The advantage is that this feature will not change with rotation or scale, so we only require a 2D space to locate the shape. The definition of Q depends on the application and the type of transformation. The most general invariant properties can be obtained by considering geometric definitions. In the case of rotation and scale changes (i.e., similarity transformations), the fundamental invariant property is given by the concept of angle.

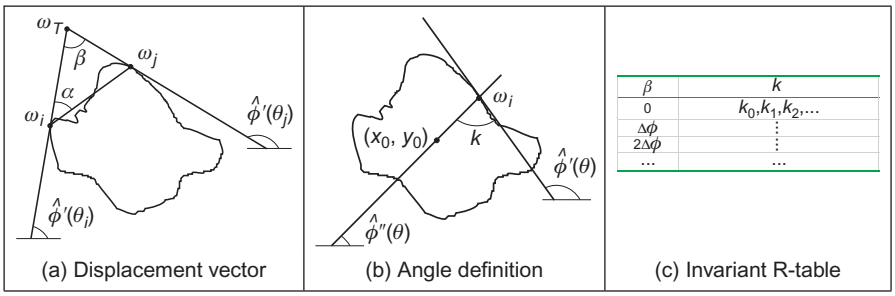


FIGURE 5.32

Geometry of the invariant GHT.

An angle is defined by three points and its value remains unchanged when it is rotated and scaled. Thus, if we associate to each edge point ω_i a set of other two points $\{\omega_j,\omega_T\}$, we can compute a geometric feature that is invariant to similarity transformations. That is,

$$Q(\omega_i) = \frac{\omega_{xj}\omega_{yi} - \omega_{xi}\omega_{yj}}{\omega_{xi}\omega_{xj} + \omega_{yi}\omega_{yj}} \tag{5.88}$$

where ω_{xn} and ω_{yn} are the x and the y coordinates of point n. Equation (5.88) defines the tangent of the angle at the point ω_T. In general, we can define the points $\{\omega_j,\omega_T\}$ in different ways. An alternative geometric arrangement is shown in Figure 5.32(a). Given the points ω_i and a fixed angle ϑ, we determine the point ω_j such that the angle between the tangent line at ω_i and the line that joins the points is ϑ. The third point is defined by the intersection of the tangent lines at ω_i and ω_j. The tangent of the angle β is defined by Eq. (5.88). This can be expressed in terms of the points and its gradient directions as

$$Q(\omega_i) = \frac{\phi'_i - \phi'_j}{1 + \phi'_i\phi'_j} \tag{5.89}$$

We can replace the gradient angle in the R-table, by the angle β. The form of the new invariant table is shown in Figure 5.32(c). Since the angle β does not change with rotation or change of scale, we do not need to change the index for each potential rotation and scale. However, the displacement vectors change according to rotation and scale (i.e., Eq. (5.85)). Thus, if we want an invariant formulation, we must also change the definition of the position vector.

In order to locate the point b, we can generalize the ideas presented in Figures 5.26(a) and 5.28(a). Figure 5.32(b) shows this generalization. As in the case of the circle and ellipse, we can locate the shape by considering a line of votes that passes through the point b. This line is determined by the value of ϕ''_i. We will do two things. First, we will find an invariant definition of this value. Secondly, we will include it on the GHT table.

We can develop Eq. (5.73) as

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} \omega_{xi} \\ \omega_{yi} \end{bmatrix} + \lambda \begin{bmatrix} \cos(\rho) & \sin(\rho) \\ -\sin(\rho) & \cos(\rho) \end{bmatrix} \begin{bmatrix} x(\theta) \\ y(\theta) \end{bmatrix} \tag{5.90}$$

Thus, Eq. (5.60) generalizes to

$$\phi''_i = \frac{\omega_{yi} - y_0}{\omega_{xi} - x_0} = \frac{[-\sin(\rho)\cos(\rho)]y(\theta)}{[\cos(\rho)\sin(\rho)]x(\theta)} \tag{5.91}$$

By some algebraic manipulation, we have

$$\phi''_i = \tan(\xi - \rho) \tag{5.92}$$

where

$$\xi = \frac{y(\theta)}{x(\theta)} \tag{5.93}$$

In order to define ϕ''_i we can consider the tangent angle at the point ω_i. By considering the derivative of Eq. (5.72), we have

$$\phi'_i = \frac{[-\sin(\rho)\cos(\rho)]y'(\theta)}{[\cos(\rho)\sin(\rho)]x'(\theta)} \tag{5.94}$$

Thus,

$$\phi'_i = \tan(\phi - \rho) \tag{5.95}$$

where

$$\phi = \frac{y'(\theta)}{x'(\theta)} \tag{5.96}$$

By considering Eqs (5.92) and (5.95), we define

$$\hat{\phi}''_i = k + \hat{\phi}'_i \tag{5.97}$$

The important point in this definition is that the value of k is invariant to rotation. Thus, if we use this value in combination with the tangent at a point, we can have an invariant characterization. In order to see that k is invariant, we solve it for Eq. (5.97). That is,

$$k = \hat{\phi}'_i - \hat{\phi}''_i \tag{5.98}$$

Thus,

$$k = \xi - \rho - (\phi - \rho) \tag{5.99}$$

That is,

$$k = \xi - \phi \tag{5.100}$$

That is independent of rotation. The definition of k has a simple geometric interpretation illustrated in Figure 5.26(b).

In order to obtain an invariant GHT, it is necessary to know for each point ω_i, the corresponding point $v(\theta)$ and then compute the value of ϕ_i''. Then evidence can be gathered by the line in Eq. (5.91). That is,

$$y_0 = \phi_i''(x_0 - \omega_{xi}) + \omega_{yi} \tag{5.101}$$

In order to compute ϕ_i'' we can obtain k and then use Eq. (5.100). In the standard tabular form, the value of k can be precomputed and stored as function of the angle β.

Code 5.12 illustrates the implementation to obtain the invariant R-table. This code is based on Code 5.10. The value of α is set to $\pi/4$ and each element of the

```
%Invariant R-Table

function T=RTableInv(entries,inputimage)

%image size
[rows,columns]=size(inputimage);

%edges
[M,Ang]=Edges(inputimage);
M=MaxSupr(M,Ang);

alfa=pi/4;
D=pi/entries;
s=0;            %number of entries in the table
t=0;
F=zeros(entries,1);  %number of entries in the row

%compute reference point
xr=0; yr=0; p=0;
for x=1:columns
   for y=1:rows
     if(M(y,x)~=0)
        xr=xr+x;
        yr=yr+y;
        p=p+1;
     end
   end
end
xr=round(xr/p);
yr=round(yr/p);

%for each edge point
for x=1:columns
  for y=1:rows
    if(M(y,x)~=0)
       %search for the second point
```

CODE 5.12

Constructing of the invariant R-table.

```
   x1=-1; y1=-1;
   phi=Ang(y,x);
   m=tan(phi-alfa);

   if(m>-1 & m<1)
       for i=3:columns
         c=x+i;
         j=round(m*(c-x)+y);
         if(j>0 & j<rows & c>0 & c<columns & M(j,c)~=0)
           x1=c ; y1=j;
           i= columns;
         end

         c=x-i;
         j=round(m*(c-x)+y);
         if(j>0 & j<rows & c>0 & c<columns & M(j,c)~=0)
           x1=c ; y1=j;
           i=columns;
         end
       end
else
    for j=3:rows
      c=y+j;
      i=round(x+(c-y)/m);
      if(c>0 & c<rows & i>0 & i< columns & M(c,i)~=0)
        x1=i ; y1=c;
        i=rows;
      end
      c=y-j;
      i=round(x+(c-y)/m);
      if(c>0 & c<rows & i>0 & i< columns & M(c,i)~=0)
        x1=i ; y1=c;
        i= rows;
      end
    end
end

if( x1~=-1)
    %compute beta
      phi=tan(Ang(y,x));
      phj= tan(Ang(y1,x1));
      if((1+phi*phj)~=0)
        beta=atan((phi-phj)/(1+phi*phj));
      else
        beta=1.57;
      end

      %compute k
      if((x-xr)~=0)
        ph=atan((y-yr)/(x-xr));
      else
        ph=1.57;
      end
      k=ph-Ang(y,x);
```

CODE 5.12

(Continued)

```
            %insert in the table
            i=round((beta+(pi/2))/D);
            if(i==0) i=1; end;

            V=F(i)+1;

        if(V>s)
            s=s+1;
            T(:,s)=zeros(entries,1);
              end;

              T(i,V)=k;
              F(i)=F(i)+1;
            end

        end %if
      end % y
    end % x
```

CODE 5.12

(Continued)

table stores a single value computed according to Eq. (5.98). The more cumbersome part of the code is to search for the point ω_j. We search in two directions from ω_i and we stop once an edge point has been located. This search is performed by tracing a line. The trace is dependent on the slope. When the slope is between -1 and $+1$, we then determine a value of y for each value of x, otherwise we determine a value of x for each value of y.

Code 5.13 illustrates the evidence-gathering process according to Eq. (5.101). This code is based on the implementation presented in Code 5.11. We use the value of β defined in Eq. (5.89) to index the table passed as parameter to the function `GHTInv`. The data k recovered from the table is used to compute the slope of the angle defined in Eq. (5.97). This is the slope of the line of votes traced in the accumulators.

Figure 5.33 shows the accumulator obtained by the implementation of Code 5.13. Figure 5.33(a) shows the template used in this example. This template was used to construct the R-table in Code 5.12. The R-table was used to accumulate evidence when searching for the piece of the puzzle in the image in Figure 5.33(b). Figure 5.33(c) shows the result of the evidence-gathering process. We can observe a peak in the location of the object. However, this accumulator contains significant noise. The noise is produced since rotation and scale change the value of the computed gradient. Thus, the line of votes is only approximated. Another problem is that pairs of points ω_i and ω_j might not be found in an image, thus the technique is more sensitive to occlusion and noise than the GHT.

```
%Invariant Generalised Hough Transform

function GHTInv(inputimage,RTable)

%image size
[rows,columns]=size(inputimage);

%table size
[rowsT,h,columnsT]=size(RTable);
D=pi/rowsT;

%edges
[M,Ang]=Edges(inputimage);
M=MaxSupr(M,Ang);

alfa=pi/4;

%accumulator
acc=zeros(rows,columns);
% for each edge point
for x=1:columns
  for y=1:rows
    if(M(y,x)~=0)
      % search for the second point
      x1=-1; y1=-1;
      phi=Ang(y,x);
      m=tan(phi-alfa);

      if(m>-1 & m<1)
        for i=3:columns
          c=x+i;
          j=round(m*(c-x)+y);
          if(j>0 & j<rows & c>0 & c<columns & M(j,c)~=0)
            x1=c ;y1=j;
            i= columns;
          end
          c=x-i;
          j=round(m*(c-x)+y);
          if(j>0 & j<rows & c>0 & c<columns & M(j,c)~=0)
            x1=c ;y1=j;
            i=columns;
          end
        end
      else
        for j=3:rows
          c=y+j;
          i=round(x+(c-y)/m);
          if(c>0 & c<rows & i>0 & i<columns & M(c,i)~=0)
            x1=i ;y1=c;
            i=rows;
          end
```

CODE 5.13

Implementation of the invariant GHT.

```
        c=y-j;
        i=round(x+(c-y)/m);
        if(c>0 & c<rows & i>0 & i<columns & M(c,i)~=0)
          x1=i ;y1=c;
          i=rows;
        end
      end
    end

  if(x1~=-1)
    %compute beta
        phi=tan(Ang(y,x));
        phj=tan(Ang(y1,x1));
        if((1+phi*phj)~=0)
          beta=atan((phi-phj)/(1+phi*phj));
        else
          beta=1.57;
        end
        i=round((beta+(pi/2))/D);
        if(i==0) i=1; end;

        %search for k
        for j=1:columnsT
          if(RTable(i,j)==0)
            j=columnsT; % no more entries
          else
            k=RTable(i,j);
            %lines of votes
            m=tan(k+Ang(y,x));
          if(m>-1 & m<1)
            for x0=1:columns
              y0=round(y+m*(x0-x));
              if(y0>0 & y0<rows)
                acc(y0,x0)=acc(y0,x0)+1;
                    end
                  end
                else
                  for y0=1:rows
                    x0= round(x+(y0-y)/m);
                    if(x0>0 & x0<columns)
                      acc(y0,x0)=acc(y0,x0)+1;
                    end
                  end
                end
              end
            end
           end
         end %if
       end % y
     end % x
```

CODE 5.13

(Continued)

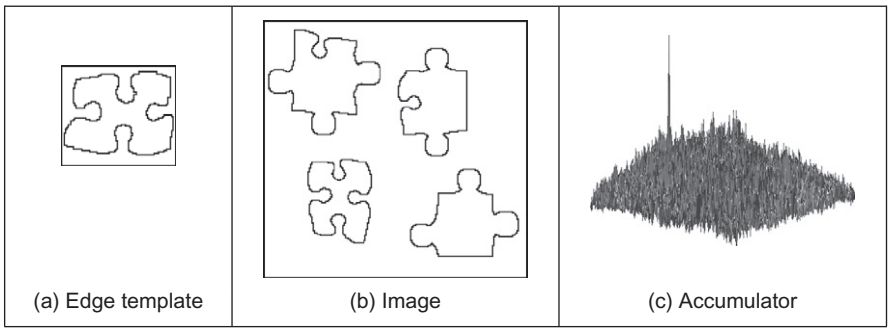


FIGURE 5.33

Applying the invariant GHT.

5.5.7 Other extensions to the HT

The motivation for extending the HT is clear: **keep** the **performance**, but **improve** the **speed**. There are other approaches to reduce the computational load of the HT. These approaches aim to improve speed and reduce memory focusing on smaller regions of the accumulator space. These approaches have included: the *fast HT* (Li and Lavin, 1986) which uses successively splits the accumulator space into quadrants and continues to study the quadrant with most evidence; the *adaptive HT* (Illingworth and Kittler, 1987) which uses a fixed accumulator size to iteratively focus onto potential maxima in the accumulator space; the *randomized HT* (Xu et al., 1990) and the *probabilistic HT* (Kälviäinen et al., 1995) which use a random search of the accumulator space; and other pyramidal techniques. One main problem with techniques which do not search the full accumulator space, but a reduced version to save speed, is that the wrong shape can be extracted (Princen et al., 1992a), a problem known as **phantom shape location**. These approaches can also be used (with some variation) to improve speed of performance in template matching. There have been many approaches aimed to improve the performance of the HT and GHT.

There has been a comparative study on the GHT (including efficiency) (Kassim et al., 1999) and alternative approaches to the GHT include two *fuzzy* HTs (Philip, 1991) which (Sonka et al., 1994) includes uncertainty of the perimeter points within a GHT structure and (Han et al., 1994) which approximately fits a shape but which requires application-specific specification of a fuzzy membership function. There have been two major reviews of the state of research in the HT (Illingworth and Kittler, 1988; Leavers, 1993) (but they are rather dated now) and a textbook (Leavers, 1992) which cover many of these topics. The analytic approaches to improving the HTs' performance use mathematical analysis to reduce size, and more importantly dimensionality, of the accumulator space. This concurrently improves speed. A review of HT-based techniques for circle extraction (Yuen et al., 1990) covered some of the most popular techniques available at the time.

5.6 Further reading

It is worth noting that much recent research has focused on shape extraction by combination of low-level features, Section 5.4, rather than on HT-based approaches. The advantages of the low-level feature approach are **simplicity**, in that the features exposed are generally less complex than the variants of the HT. There is also a putative advantage in **speed**, in that simpler approaches are invariably faster than those which are more complex. Any advantage in respect of **performance** in noise and occlusion is yet to be established. The HT approaches do not (or not yet) include machine learning approaches, which is perhaps where the potency is achieved by the techniques which use low-level features. The use of machine learning also implies a need for training, but there is a need to generate some form of template for the HT or template approaches. An overarching premise of this text is that there is no panacea and as such there is a selection of techniques as there are for feature extraction, and some of the major approaches have been covered in this chapter.

In terms of **performance evaluation**, it is worth noting the PASCAL Visual Object Classes (VOC) challenge (Everingham et al., 2010) which is a new benchmark in visual object category recognition and detection. The PASCAL consortium http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/index.html aims to provide standardized databases for object recognition; to provide a common set of tools for accessing and managing the database annotations; and to conduct challenges which evaluate performance on object class recognition. This provides evaluation data and mechanisms and thus describing many recent advances in recognizing objects from a number of visual object classes in realistic scenes.

The majority of further reading in finding shapes concerns papers, many of which have already been referenced, especially in the newer techniques. An excellent survey of the techniques used for feature extraction (including template matching, deformable templates, etc.) can be found in Trier et al. (1996). Few of the textbooks devote much space to shape extraction except *Shape Classification and Analysis* (Costa and Cesar, 2009) and *Template Matching Techniques in Computer Vision* (Brunelli, 2009), sometimes dismissing it in a couple of pages. This rather contrasts with the volume of research there has been in this area, and the HT finds increasing application as computational power continues to increase (and storage cost reduces). Other techniques use a similar evidence-gathering process to the HT. These techniques are referred to as geometric hashing and clustering techniques (Stockman, 1987; Lamdan et al., 1988). In contrast with the HT, these techniques do not define an analytic mapping, but they gather evidence by grouping a set of features computed from the image and from the model.

Essentially, this chapter has focused on shapes which can in some form have a fixed appearance whether it is exposed by a template, a set of keypoints, or by a description of local properties. In order to extend the approaches to shapes with a less constrained description, and rather than describe such shapes by constructing a library of their possible appearances, we require techniques for deformable shape analysis, as we shall find in the next chapter.

5.7 References

Aguado, A.S., 1996. Primitive Extraction via Gathering Evidence of Global Parameterised Models, Ph.D. Thesis, University of Southampton.

Aguado, A.S., Montiel, E., Nixon, M.S., 1996. On using directional information for parameter space decomposition in ellipse detection. Pattern Recog. 28 (3), 369–381.

Aguado, A.S., Nixon, M.S., Montiel, M.E., 1998. Parameterising arbitrary shapes via Fourier descriptors for evidence-gathering extraction. Comput. Vision Image Understand. 69 (2), 202–221.

Aguado, A.S., Montiel, E., Nixon, M.S., 2000a. On the intimate relationship between the principle of duality and the Hough transform. Proc. Roy. Soc. A 456, 503–526.

Aguado, A.S., Montiel, E., Nixon, M.S., 2000b. Bias error analysis of the generalised Hough transform. J. Math. Imag. Vision 12, 25–42.

Altman, J., Reitbock, H.J.P., 1984. A fast correlation method for scale- and translation-invariant pattern recognition. IEEE Trans. PAMI 6 (1), 46–57.

Arbab-Zavar, B., Nixon, M.S., 2011. On guided model-based analysis for ear biometrics. Comput. Vision Image Understand. 115, 487–502.

Ballard, D.H., 1981. Generalising the Hough transform to find arbitrary shapes. CVGIP 13, 111–122.

Bay, H., Eas, A., Tuytelaars, T., Van Gool, L., 2008. Speeded-up robust features (SURF). Comput. Vision Image Understand. 110 (3), 346–359.

Bracewell, R.N., 1986. The Fourier Transform and its Applications, second ed. McGraw-Hill, Singapore.

Bresenham, J.E., 1965. Algorithm for computer control of a digital plotter. IBM Syst. J. 4 (1), 25–30.

Bresenham, J.E., 1977. A linear algorithm for incremental digital display of circular arcs. Comms. ACM 20 (2), 750–752.

Brown, C.M., 1983. Inherent bias and noise in the Hough transform. IEEE Trans. PAMI 5, 493–505.

Brunelli, R., 2009. Template Matching Techniques in Computer Vision. Wiley, Chichester.

Bustard, J.D., Nixon, M.S., 2010. Toward unconstrained ear recognition from two-dimensional images. IEEE Trans SMC(A) 40 (3), 486–494.

Casasent, D., Psaltis, D., 1977. New optical transforms for pattern recognition. Proc. IEEE 65 (1), 77–83.

Costa, L.F., Cesar, L.M., 2009. Shape Classification and Analysis, second ed. CRC Press and Taylor & Francis, Boca Raton, FL.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. Proc. IEEE Conf. Comput. Vision Pattern Recog. 2, 886–893.

Datta, R., Joshi, D., Li, J., Wang, J.Z., 2008. Image retrieval: ideas, influences, and trends of the new age. ACM Comput. Surv. 40 (2), Article 5.

Deans, S.R., 1981. Hough transform from the radon transform. IEEE Trans. PAMI 13, 185–188.

Duda, R.O., Hart, P.E., 1972. Use of the Hough transform to detect lines and curves in pictures. Comms. ACM 15, 11–15.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The PASCAL Visual Object Classes (VOC) challenge. Int. J. Comput. Vision 88 (2), 303–338.

Gerig, G., Klein, F., 1986. Fast contour identification through efficient Hough transform and simplified interpretation strategy. Proceedings of the Eighth International Conference on Pattern Recognition, pp. 498–500.

Grimson, W.E.L., Huttenglocher, D.P., 1990. On the sensitivity of the Hough transform for object recognition. IEEE Trans. PAMI 12, 255–275.

Han, J.H., Koczy, L.T., Poston, T., 1994. Fuzzy Hough transform. Pattern Recog. Lett. 15, 649–659.

Hecker, Y.C., Bolle, R.M., 1994. On geometric hashing and the generalized Hough transform. IEEE Trans. SMC 24, 1328–1338.

Hough, P.V.C., 1962. Method and Means for Recognising Complex Patterns, US Patent 3069654.

Hurley D.J., Arbab-Zavar, B., Nixon, M.S., 2008. The ear as a biometric. In: Jain, A., Flynn, P., Ross, A. (Eds.), Handbook of Biometrics, pp. 131–150.

Illingworth, J., Kittler, J., 1987. The adaptive Hough transform. IEEE Trans. PAMI 9 (5), 690–697.

Illingworth, J., Kittler, J., 1988. A survey of the Hough transform. CVGIP 48, 87–116.

Kälviäinen, H., Hirvonen, P., Xu, L., Oja, E., 1995. Probabilistic and non-probabilistic Hough transforms: overview and comparisons. Image Vision Comput. 13 (4), 239–252.

Kassim, A.A., Tan, T., Tan, K.H., 1999. A comparative study of efficient generalised Hough transform techniques. Image Vision Comput. 17 (10), 737–748.

Kimme, C., Ballard, D., Sklansky, J., 1975. Finding circles by an array of accumulators. Comms. ACM 18 (2), 120–122.

Kiryati, N., Bruckstein, A.M., 1991. Antialiasing the Hough transform. CVGIP Graph. Models Image Process. 53, 213–222.

Lamdan, Y., Schawatz, J., Wolfon, H., 1988. Object recognition by affine invariant matching. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 335–344.

Leavers, V., 1992. Shape Detection in Computer Vision Using the Hough Transform. Springer-Verlag, London.

Leavers, V., 1993. Which Hough transform, CVGIP: Image Understand., **58**.

Li, H., Lavin, M.A., 1986. Fast Hough transform: a hierarchical approach. CVGIP 36, 139–161.

Lienhart, R., Kuranov, A., Pisarevsky, V., 2003. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. LNCS 2781, 297–304.

Lowe, D.G., 2004. Distinctive image features from scale-invariant key points. Int. J. Comput. Vision 60 (2), 91–110.

Merlin, P.M., Farber, D.J., 1975. A parallel mechanism for detecting curves in pictures. IEEE Trans. Computers 24, 96–98.

Mikolajczyk, K., Schmid, C., 2005. A performance evaluation of local descriptors. IEEE Trans. PAMI 27 (10), 1615–1630.

O'Gorman, F., Clowes, M.B., 1976. Finding picture edges through collinearity of feature points. IEEE Trans. Computers 25 (4), 449–456.

Philip, K.P., 1991. Automatic Detection of Myocardial Contours in Cine Computed Tomographic Images, Ph.D. Thesis, Iowa University.

Princen, J., Yuen, H.K., Illingworth, J., Kittler, J., 1992a. Properties of the adaptive Hough transform. Proceedings of the Sixth Scandinavian Conference on Image Analysis, Oulu, Finland.

Princen, J., Illingworth, J., Kittler, J., 1992b. A formal definition of the Hough transform: properties and relationships. J. Math. Imag. Vision 1, 153–168.

Rosenfeld, A., 1969. Picture Processing by Computer. Academic Press, London.

Schneiderman, H., Kanade, T., 2004. Object detection using the statistics of parts. Int. J. Comput. Vision 56 (3), 151–177.

Sivic, J., Zisserman, A., Video Google, A, 2003. Text retrieval approach to object matching in videos. Proc. IEEE ICCV'03 2, 1470–1477.

Sklansky, J., 1978. On the Hough technique for curve detection. IEEE Trans. Computers 27, 923–926.

Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R., 2000. Content-based image retrieval at the end of the early years. IEEE Trans. PAMI 22 (12), 1349–1378.

Sonka, M., Hllavac, V., Boyle, R., 1994. Image Processing, Analysis and Computer Vision. Chapman Hall, London.

Stockman, G., 1987. Object recognition and localization via pose clustering. CVGIP 40, 361–387.

Stockman, G.C., Agrawala, A.K., 1977. Equivalence of Hough curve detection to template matching. Comms. ACM 20, 820–822.

Traver, V.J., Pla, F., 2003. The log-polar image representation in pattern recognition tasks. Lect. Notes Comput. Sci. 2652, 1032–1040.

Trier, O.D., Jain, A.K., Taxt, T., 1996. Feature extraction methods for character recognition—a survey. Pattern Recog. 29 (4), 641–662.

Tuytelaars, T., Mikolajczyk, K., 2007. Local invariant feature detectors: a survey. Found. Trends Comput. Graphics Vision 3 (3), 177–280.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. Proc. IEEE Conf. Computer Vision Pattern Recog. 1, 511–519.

Viola, P., Jones, M.J., 2004. Robust real-time face detection. Int. J. Comput. Vision 57 (2), 137–154.

Xu, L., Oja, E., Kultanen, P., 1990. A new curve detection method: randomised Hough transform. Pattern Recog. Lett. 11, 331–338.

Yuen, H.K., Princen, J., Illingworth, J., Kittler, J., 1990. Comparative study of Hough transform methods for circle finding. Image Vision Comput. 8 (1), 71–77.

Zhu, Q., Avidan, S., Yeh, M.-C., Cheng, K.-T., 2006. Fast human detection using a cascade of histograms of oriented gradients. Proc. IEEE Conf. Computer Vision Pattern Recog. 2, 1491–1498.

Zokai, S., Wolberg, G., 2005. Image registration using log-polar mappings for recovery of large-scale similarity and projective transformations. IEEE Trans. IP 14, 1422–1434.