CHAPTER

Line, circle, and ellipse detection

10

Detection of macroscopic features is an important aspect of visual pattern recognition. Of particular interest is the identification of straight lines, which are ubiquitous both in manufacturing and in the built environment. Two methods—the Hough transform (HT) and RANSAC (random sample consensus)—provide the means for locating these features highly robustly. This chapter describes the principles and processes needed to achieve this. In addition, circle features occur very widely in images of manufactured objects and can also be located using HTs. This also applies for ellipses, which may appear in their own right or as oblique views of circular objects. However, locating ellipses is more complicated than locating circles because of the greater number of parameters needed to define them.

Look out for:

- how HTs may be used for locating straight lines in images
- why HTs are robust against noise and background clutter
- the RANSAC approach to line location, and its efficiency
- how laparoscopic tools may be located
- how HTs may be used for locating circular objects in images
- how they can be applied when circle radius is unknown
- how speed of processing can be increased
- the diameter bisection and chord−tangent methods for ellipse location
- how the human iris may be located.

This chapter describes two key methods for detecting important features in images. It is shown how the HT is able to locate a range of features including straight lines, circles, and ellipses, while the RANSAC approach is applied to the efficient location of straight lines, though its power greatly exceeds this single illustration. Both methods rely on voting schemes, and in each case their robustness stems from concentration on *positive* evidence for objects and features.

10.1 INTRODUCTION

Straight edges are amongst the most common features of the modern world, arising in perhaps the majority of manufactured objects and components—not

Computer Vision. DOI: http://dx.doi.org/10.1016/B978-0-12-809284-2.00010-1

least in the very buildings in which we live. Yet, it is arguable whether true straight lines ever arise in the natural state: possibly the only example of their appearance in virgin outdoor scenes is the horizon—although even this is clearly seen from space as a circular boundary! The surface of water is essentially planar, although it is important to realize that this is a deduction: the fact remains that straight lines seldom appear in completely natural scenes. Be all this as it may, it is clearly vital both in city pictures and in the factory to have effective means of detecting straight edges. This chapter studies available methods for locating these important features.

Historically, the HT has been the main means of detecting straight edges, and since the method was originally invented (Hough, 1962), it has been developed and refined for this purpose. Hence this chapter concentrates on this particular technique; it also prepares the ground for applying the HT to the detection of circles, ellipses, corners, etc. in the next few chapters. We start by examining the original Hough scheme, even though it is now seen to be wasteful in computation, since important principles are involved. By the end of the chapter we shall see that the HT is not alone in its capabilities for line detection: RANSAC is also highly capable in this direction. In fact, both approaches have their advantages and limitations, as the discussion in Section 10.6 will show.

This chapter also examines the location of round objects: these are important in many areas of image analysis but they are especially important in industrial applications such as automatic inspection and assembly. In the food industry alone, a very sizeable number of products are round—biscuits, cakes, pizzas, pies, oranges, and so on (Davies, 1984c). In the automotive industry many circular components are used—washers, wheels, pistons, heads of bolts, etc., while round holes of various sizes appear in such items as casings and cylinder blocks. In addition, buttons and many other everyday objects are round. Of course, when round objects are viewed obliquely they appear elliptical; furthermore, certain other objects are *actually* elliptical. This makes it clear that we need algorithms that are capable of finding both circles and ellipses. Finally, objects can frequently be located by their holes, so finding round holes or features is part of the larger problem: this chapter addresses various aspects of this problem.

An important facet of this work is how well object location algorithms cope in the presence of artifacts such as shadows and noise. In particular, the paradigm represented by Table 10.1 was shown in Chapter 9, Boundary Pattern Analysis to be insufficiently robust to cope in such situations. This chapter shows that the HT technique is particularly good at dealing with all sorts of difficulties, including quite severe occlusions. It achieves this not by adding robustness but by having robustness built in as an integral part of the technique.

The application of the HT to circle detection is one of the most straightforward uses of the technique. However, there are several enhancements and adaptations that can be applied in order to improve accuracy and speed of operation and in addition to make the method work efficiently when detecting circles with a range of sizes. These modifications are studied after covering the basic HT

Table 10.1 Basic RANSAC Algorithm for Finding the Line With Greatest Support

```
Mmax = 0;
for all pairs of edge points do {
  find equation of line defined by the two points i, j;
  M = 0;
  for all N points in list do
    if (point k is within threshold distance d of line) M++;
  if (M > Mmax) {
    Mmax = M;
    imax = i;
    jmax = j;
    // this records the hypothesis giving the maximum support so far
  }
}
/* if Mmax > 0, (x[imax], y[imax]) and (x[jmax], y[jmax]) will be the
coordinates of the points defining the line having greatest support */
```

This algorithm only returns one line: in fact it returns the specific line model that has greatest support, for the line that has greatest support. Lines with less support are in the end ignored.

techniques. Versions of the HT that can perform ellipse detection are then considered. The chapter ends with a short section on an important application—that of human iris location.

10.2 APPLICATION OF THE HOUGH TRANSFORM TO LINE DETECTION

The basic concept involved in locating lines by the HT is point-line duality. A point P can be defined either as a pair of coordinates or in terms of the set of lines passing through it. The concept starts to make sense if we consider a set of collinear points P_i, then list the sets of lines passing through each of them, and finally note that there is just one line that is common to all these sets. Thus it is possible to find the line containing all the points P_i merely by eliminating those that are not multiple hits. Indeed, it is easy to see that if a number of noise points Q_j are intermingled with the signal points P_i, the method will be able to discriminate the collinear points from amongst the noise points at the same time as finding the line containing them, merely by searching for multiple hits. Thus the method is inherently robust against noise, as indeed it is in discriminating against currently unwanted signals such as circles.

In fact, the duality goes further. For just as a point can define (or be defined by) a set of lines, so a line can define (or be defined by) a set of points, as is obvious from the above argument. This makes the above approach to line detection a mathematically elegant one and it is perhaps surprising that the

Hough detection scheme was first published as a patent (Hough, 1962) of an electronic apparatus for detecting the tracks of high-energy particles, rather than as a paper in a learned journal.

The form in which the method was originally applied involves parametrizing lines using the slope-intercept equation

$$y = mx + c \tag{10.1}$$

Every point on a straight edge is then plotted as a line in (m, c) space corresponding to all the (m, c) values consistent with its coordinates, and lines are detected in this space. The embarrassment of unlimited ranges of the (m, c) values (near-vertical lines require near-infinite values of these parameters) is overcome by using two sets of plots, the first corresponding to slopes of less than 1.0 and the second to slopes of 1.0 or more; in the latter case, Eq. (10.1) is replaced by the form:

$$x = \tilde{m}x + \tilde{c} \tag{10.2}$$

where

$$\tilde{m} = 1/m \tag{10.3}$$

The need for this rather wasteful device was removed by the Duda and Hart (1972) approach, which replaces the slope-intercept formulation with the so-called "normal" (θ, ρ) form for the straight line (see Fig. 10.1):

$$\rho = x \cos\theta + y \sin\theta \tag{10.4}$$

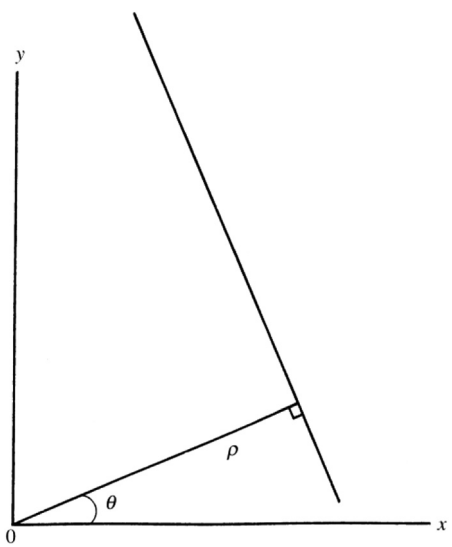


FIGURE 10.1

Normal (θ, ρ) parametrization of a straight line.

To apply the method using this form, the set of lines passing through each point P_i is represented as a set of sine curves in (θ, ρ) space: e.g., for point $P_1(x_1, y_1)$ the sine curve has equation:

$$\rho = x_1 \cos\theta + y_1 \sin\theta \tag{10.5}$$

Then multiple hits in (θ, ρ) space indicate, via their θ, ρ values, the presence of lines in the original image.

Each of the methods described above has the feature that it employs an "abstract" parameter space in which multiple hits are sought. Above we talked about "plotting" points in parameter space but in fact the means of looking for hits is to seek peaks which have been built by *accumulation* of data from various sources. Although it might be possible to search for hits by logical operations such as use of the logical AND function, the Hough method gains considerably by *accumulating evidence* for events by a *voting scheme*. It will be seen below that this is the source of the method's high degree of robustness.

Although the methods described above are mathematically elegant and are capable of detecting lines (or sets of collinear points—which may be completely isolated from each other) amid considerable interfering signals and noise, they are subject to considerable computational problems. The reason for this is that every prominent point in the original image gives rise to a great many votes in parameter space, so for a 256×256 image the (m, c) parametrization requires 256 votes to be accumulated, while the (θ, ρ) parametrization requires a similar number—360 if the θ quantization is to be fine enough to resolve 1° changes in line orientation. (It should be remarked that it does not matter in what way the "prominent points" are prominent: they may in fact be edge points, dark specks, centers of holes, and so on. Later we shall consistently take them to be edge points.)

Several workers tried to overcome this problem, and Dudani and Luk (1978) tackled it by trying to separate out the θ and ρ estimations. They accumulated votes first in a 1-D parameter space for θ—i.e., a histogram of θ values: note that such a histogram is itself a simple form of HT. (Indeed, it is now common for any process to be called an HT if it involves accumulating votes in a parameter space, with the intention of searching for significant peaks to find properties of the original data.) Having found suitable peaks in the θ histogram, they then built a ρ histogram for all the points that contributed votes to a given θ peak, and repeated this for all θ peaks. Thus two 1-D spaces replace the original 2-D parameter space, with very significant savings in storage and load. However, two-stage methods of this type tend to be less accurate since the first stage is less selective: biased θ values may result from pairs of lines that would be well separated in a 2-D space. In addition, any error in estimating θ values is propagated to the ρ determination stage, making values of ρ even less accurate. For this reason Dudani and Luk added a final least-squares fitting stage to complete the accurate analysis of straight edges present in the image. (Note that many workers have found that using the least-squares technique tends to weight up the contribution of less accurate points, including those that have nothing to do with the line in

question. To understand the limitations of least squares, see the Appendix on Robust Statistics.)

From a practical point of view, to proceed with this method of line detection, it is first necessary to obtain the local components of intensity gradient, and then deduce the gradient magnitude g and threshold it to locate each edge pixel in the image. θ may be estimated using the arctan function in conjunction with the local edge gradient components g_x, g_y:

$$\theta = \arctan(g_y/g_x) \tag{10.6}$$

As the arctan function has period π, $\pm\pi$ may have to be added to obtain a principal value in the range $-\pi$ to $+\pi$: this can be decided from the signs of g_x and g_y. (Note that in C++, the basic arctan function is *atan*, with a single argument, which should be g_y/g_x used as indicated above. However, the C++ *atan2* function has two arguments, and if g_y and g_x are used, respectively, for these, the function automatically returns an angle in the range $-\pi$ to π.) Once θ is known, ρ can be found from Eq. (10.4).

Finally, note that straight lines and straight edges are different and need to be detected differently. (Straight *edges* are probably more common and appear as object boundaries, whereas straight *lines* are typified by telephone wires in outdoor scenes.) In fact, we have concentrated above on using the HT to locate straight edges, starting with edge detectors. Straight line segments may be located using Laplacian-type operators and their orientations are defined over a range 0° to 180° rather than 0° to 360°: this makes HT design subtly different. For concreteness, in the remainder of this chapter we concentrate on straight *edge* detection.

10.2.1 LONGITUDINAL LINE LOCALIZATION

The preceding sections have provided a variety of means for locating lines in digital images and finding their orientations. However, these methods are insensitive to where along an infinite idealized line an observed segment appears. The reason for this is that the fit includes only two parameters. There is some advantage to be gained in this, in that partial occlusion of a line does not prevent its detection: indeed, if several segments of a line are visible, they can all contribute to the peak in parameter space, hence improving sensitivity. On the other hand, for full image interpretation, it is useful to have information about the longitudinal placement of line segments.

This is achieved by a further stage of processing. The additional stage involves finding which points contributed to each peak in the main parameter space, and carrying out connectivity analysis in each case. (When the slope of the line is less than 45°, this is most conveniently achieved by projecting it along the x-axis, and when greater than 45°, by projecting it along the y-axis: similarly for orientations in other quadrants.) Dudani and Luk (1978) called this process "xy-grouping." It is not vital that the line segments should be 4- or 8-connected—just that

there should be sufficient points on them so that adjacent points are within a threshold distance apart, i.e., groups of points are merged if they are within the prespecified distance (typically, 5 pixels). Finally, segments shorter than a certain minimum length (also typically $\sim$5 pixels) can be ignored as too insignificant to help with image interpretation.

10.3 THE FOOT-OF-NORMAL METHOD

An alternative means of saving computation (Davies, 1986) eliminates the use of trigonometric functions such as arctan by employing a different parametrization scheme. As noted earlier, the methods so far described all employ abstract parameter spaces in which points bear no immediately obvious visual relation to image space. In the alternative scheme, the parameter space is a second image space, which is congruent to image space (i.e., parameter space is like image space, *and* each point in parameter space holds information that is immediately relevant to the corresponding point in image space).

This type of parameter space is obtained in the following way. First, each edge fragment in the image is produced much as required previously so that ρ can be measured, but this time the foot of the normal from the origin is itself taken as a voting position in parameter space (Fig. 10.1). Clearly, the foot-of-normal position embodies all the information previously carried by the ρ and θ values, and mathematically the methods are essentially equivalent. However, the details differ, as will be seen.

The detection of straight edges starts with the analysis of (1) local pixel coordinates (x, y) and (2) the corresponding local components of intensity gradient (g_x, g_y) for each edge pixel. Taking (x_0, y_0) as the foot of the normal from the origin to the relevant line (produced if necessary—see Fig. 10.2), it is found that

$$g_y/g_x = y_0/x_0 \tag{10.7}$$

$$(x - x_0)x_0 + (y - y_0)y_0 = 0 \tag{10.8}$$

These two equations are sufficient to compute the two coordinates (x_0, y_0). Solving for x_0 and y_0 gives

$$x_0 = vg_x \tag{10.9}$$

$$y_0 = vg_y \tag{10.10}$$

where

$$v = \frac{xg_x + yg_y}{g_x^2 + g_y^2} \tag{10.11}$$

Notice that these expressions involve only additions, multiplications, and just one division, so voting can be carried out efficiently using this formulation.

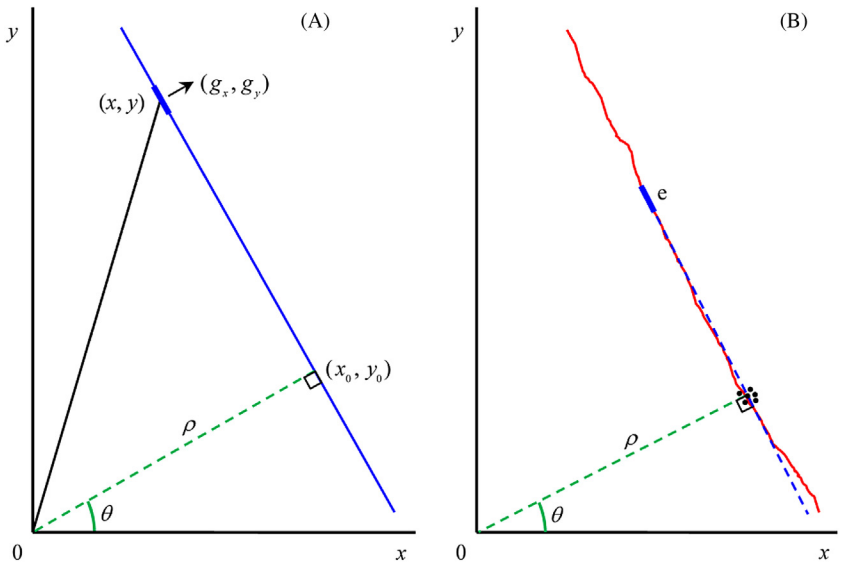


FIGURE 10.2

Image space parametrization of a straight line: (A) parameters involved in the calculation (see text); (B) buildup of foot-of-normal positions in parameter space for a more practical situation, where the line is not exactly straight: e is a typical edge fragment leading to a single vote in parameter space.

10.3.1 APPLICATION OF THE FOOT-OF-NORMAL METHOD

Although the foot-of-normal method is mathematically similar to the (θ, ρ) method, it is unable to determine line orientation directly with quite the same degree of accuracy. This is because the orientation accuracy depends on the fractional accuracy in determining ρ—which in turn depends on the absolute magnitude of ρ. Hence for small ρ the orientation of a line that is predicted from the position of the peak in parameter space will be relatively inaccurate, even though the *position* of the foot-of-normal is known accurately. However, accurate values of line orientation can always be found by identifying the points that contributed to a given peak in the foot-of-normal parameter space and making them contribute to a θ histogram, from which line orientation may be determined much more accurately.

Typical results with the above method are shown in Fig. 10.3: here it was applied in subimages of size 64×64 within 128×128 images. Clearly, some of the objects in these pictures are grossly overdetermined by their straight edges, so low ρ values are not a major problem. For those peaks where $\rho > 10$, line orientation is estimated within approximately 2°; as a result, these objects are located within 1 pixel and orientated within 1° by this technique, without the necessity for θ histograms. Two of the subimages in Fig. 10.3 contain line segments that

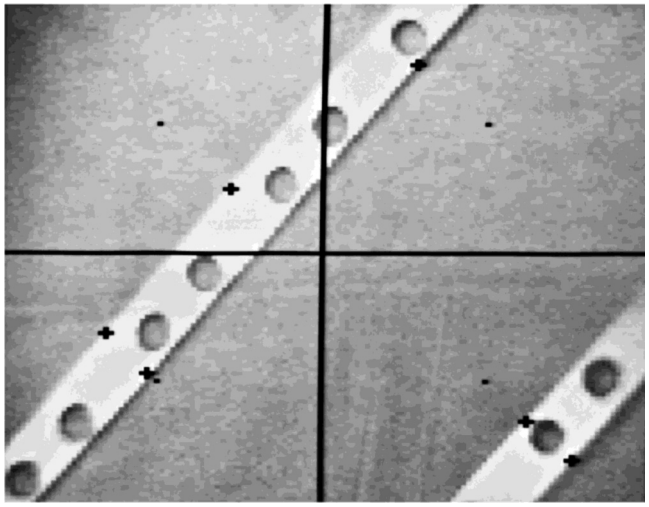


FIGURE 10.3

Results of image space parametrization of mechanical parts. The dot at the center of each quadrant is the origin used for computing the image space transform. The crosses are the positions of peaks in parameter space which mark the individual straight edges. For further explanation, see text.

are not detected. This is due partly to their relatively low contrast, higher noise levels, above average fuzziness, or short length. However, it is also due to the thresholds set on the initial edge detector and on the final peak detector: when these are set at lower values, additional lines are detected but other noise peaks also become prominent in parameter space, and each of these needs to be checked in detail to confirm the presence of the corresponding line in the image. This is one aspect of a general problem that arises right through the field of image analysis.

10.4 USING RANSAC FOR STRAIGHT LINE DETECTION

RANSAC is an alternative model-based search schema that can often be used instead of the HT. In fact, it is highly effective when used for line detection, which is why the method is introduced here. The strategy can be construed as a voting scheme, but it is used in a different way from that in the HT. The latter operates by building up the evidence for instances of target objects in the form of votes in parameter space, and then making decisions about their existence (or by making hypotheses about their existence that can subsequently be checked out). RANSAC operates by making a sequence of hypotheses about the target objects, and determines the support for each of them by counting how many data points agree with them. As might be expected, for any potential target object,

only the hypotheses with the maximum support are retained at each stage. This results in more compact information storage than for the HT: i.e., for RANSAC a list of hypotheses is held in current memory, whereas for the HT a whole parameter space, which is usually only sparsely populated, is held in memory. Thus the RANSAC data are abstract lists, whereas the HT data can often be viewed as pictures in parameter space—as in the case of the foot-of-normal line detector. None of this prevents the RANSAC output being displayed (e.g., as straight lines) in image space; nor does it prevent the HT being accumulated using a list representation.

To explain RANSAC in more detail, we take the case of line detection. As for the HT, we start by applying an edge detector and locating all the edge points in the image. As we shall see, RANSAC operates best with a limited number of points, so it is useful to find the edge points that are local maxima of the intensity gradient image. (This does not correspond to the type of nonmaximum suppression used in the Canny operator, which produces thin connected *strings* of edge points, but to individual *isolated* points: we shall return to this point below.) Next, to form a straight line hypothesis, all that is necessary is to take any pair of edge points from the list of N that remain after applying the local maximum operation. For each hypothesis we run through the list of N points finding how many points M support the hypothesis. Then we take more hypotheses (more pairs of edge points) and at each stage retain only the one giving maximum support $M_{\max}$. This process is shown in Table 10.1.

The algorithm in Table 10.1 corresponds to finding the center of the highest peak in parameter space in the case of the HT. To find all the lines in the image, the most obvious strategy is the following: find the first line, then eliminate all the points that gave it support; then find the next line and eliminate all the points that gave it support; and so on until all the points have been eliminated from the list. The process may be written more compactly in the form:

```
repeat {
  find line;
  eliminate support;
}
until no data points remain;
```

Such a strategy carries the problem that if lines cross each other, support for the second line (which will necessarily be the weaker one) could have less support than it deserves. However, this should only be a serious disadvantage if the image is severely cluttered with lines. Nevertheless, the process is sequential and as such the results (i.e., the exact line locations) will depend on the order in which lines are eliminated, as the support regions will be minutely altered at each stage. Overall, the interpretation of complex images almost certainly has to proceed sequentially, and there is significant evidence that the human eye-brain system interprets images in this way, following early cues in order to progressively make sense of the data. Interestingly, the HT seems to escape from this by the potential capability for *parallel* identification of peaks: while for simple images this may

well be true, for complicated images containing many overlapping edges, there will again be the need for sequential analysis of the type envisaged above (e.g., see the "back-projection" method of Gerig and Klein, 1986). The point is that with the particular list representation employed by RANSAC, we are *immediately* confronted with the problem of how to identify multiple targets, whereas for the reasons given above this doesn't immediately happen with the HT. Finally, on the plus side, successive elimination of support points necessarily makes it progressively easier and less computation intensive to find subsequent target objects. But the process itself is not cost-free, as the whole RANSAC procedure in Table 10.1 has to be run twice per line in order to identify the support points that have to be eliminated.

Next we consider the computational load of the RANSAC process. If there are N edge points, the number of potential lines will be ${}^{N}C_2$, corresponding to a computational load of $O(N^2)$. However, finding the support for each line will involve $O(N)$ operations, so the overall computational load will be $O(N^3)$. In addition, the need to eliminate the support points for each line found will require computation proportional to the number of lines n, amounting to only $O(nN)$, and this will have little effect on the overall computational load.

One point that has not yet been made is that all N edge points will not arise from straight lines: some will arise from lines, some from curves, some from general background clutter, and some from noise. To limit the number of false positives, it will be useful to set a support threshold M_{thr} such that potential lines for which $M > M_{\text{thr}}$ are most likely to be true straight lines, while others are most likely to be artifacts, such as parts of curves or noise points. Thus the RANSAC procedure can be terminated when $M_{\max}$ drops below M_{thr}. Of course, it may be required to retain only "significant" lines, e.g., those having length greater than L pixels. In that case, analysis of each line could allow many more points to be eliminated as the RANSAC algorithm proceeds. Another factor is whether hypotheses corresponding to pairs whose points are too close together should be taken into account. In particular, it might be considered that points closer together than 5 pixels would be superfluous as they would be likely to have much reduced chance of pointing along the direction of a line. However, it turns out that RANSAC is fail-safe in this respect, and there is some gain from keeping pairs with quite small separations, as some of the resulting hypotheses can actually be more accurate than any others. Overall, restricting pairs by their separations can be a useful way of reducing computational load, bearing in mind that $O(N^3)$ is rather high. Here, we should recall that the load for the HT is $O(N^2)$ during voting, if pairs of points are used, or $O(N)$ if single edge points and their gradients are used instead.

As we have just seen, RANSAC does not compare well with the HT regarding computational load, so it is better to employ RANSAC when N can be reduced in some way. This is why it is useful to use N local maxima rather than a full list of edge points in the form of strings of edge points generated by nonmaximum suppression or *a fortiori* those existing before nonmaximum suppression. Indeed,

there is much to be gained by repeated random sampling from the full list until sufficient hypotheses have been tested to be confident that all significant lines have been detected. [Note that these ideas reflect the original meaning of the term RANSAC, which stands for RANdom SAmpling Consensus—"consensus" indicating that any hypothesis has to form a consensus with the available support data.] Using this procedure, the computational load is reduced from $O(N^3)$ toward $O(N^2)$ or even $O(N)$ (it is difficult to predict the resulting computational complexity: in any case, the achievable computational load will be highly data dependent). Confidence that all significant lines have been detected can be obtained by estimating the risk that a significant line will be missed because no representative pair of points lying on the line was considered. This aspect of the problem will be examined more fully in Section A.6 of the Appendix on Robust Statistics.

Before proceeding further, we shall briefly consider another way of reducing computational load that is, by using the connected strings of edge points resulting from nonmaximum suppression, but in addition eliminating those that are very short and coding the longer ones as isolated points every p pixels, where $p \approx 10$. In this way, there would be far fewer points than for our earlier paradigm, and also those that are employed would have increased coherence and probability of lying on straight edges; hence there would be a concentration on high quality data, whereas the $O(N^3)$ computation factor would be dramatically reduced. Clearly, in high noise situations this would not work well: it is left to the reader to judge how well it would work for the sets of edges located by the Canny operator in Figs. 5.7 and 5.8.

We are now in a position to consider actual results obtained by applying RANSAC to straight line detection. In the tests described, pairs of points were employed as hypotheses, and all edge points were local maxima of the intensity gradient. The cases shown in Fig. 10.4 correspond to detection of a block of wood in the shape of an icosahedron, and a pair of laparoscopic tools with parallel sides. Note that one line on the right of Fig. 10.4A was missed because a lower limit had to be placed on the level of support for each line: this was necessary because below this level of support the number of chance collinearities rose dramatically even for the relatively small number of edge points shown in Fig. 10.4B, leading to a sharp rise in the number of false positive lines. Figs. 23.2 and 23.3 show RANSAC being used to locate road lane markings. The same version of RANSAC was used in all the above cases, albeit in the case of Fig. 23.3 a refinement was added to allow improved elimination of points on lines that had already been located (see below). Overall, this set of examples shows that RANSAC is a highly important contender for location of straight lines in digital images. Not discussed here is the fact that RANSAC is useful for obtaining robust fits to many other types of shape, in 2-D and in 3-D.

It should be mentioned that one characteristic of RANSAC is that it is less influenced by aliasing along straight lines than the HT. This is because HT peaks tend to be fragmented by aliasing, so the best hypotheses can be difficult to obtain without excessive smoothing of the image. The reason why RANSAC wins in

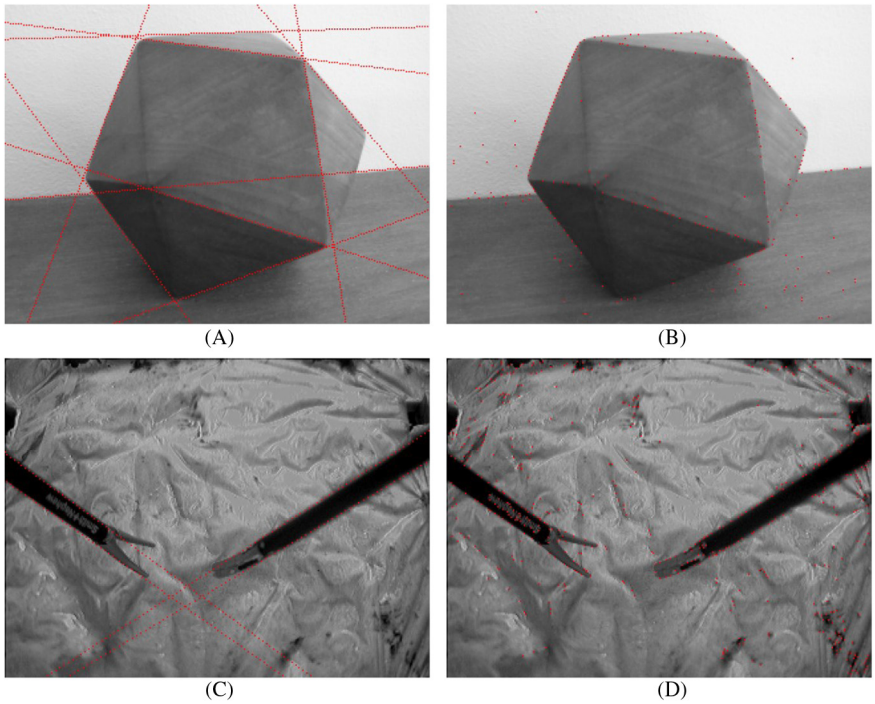


FIGURE 10.4

Straight line location using the RANSAC technique. (A) shows an original gray-scale image with various straight edges located using the RANSAC technique. (B) shows the edge points fed to RANSAC for (A). These were isolated points which were local maxima of the gradient image. (C) shows the straight edges of a pair of laparoscopic tools—a cutter and a gripper—which have been located by RANSAC. (D) shows the points fed to RANSAC for (C). In (A), three edges of the icosahedron are missed. This is because they are roof edges with low contrast and low intensity gradient. RANSAC missed a fourth edge because of a lower limit placed on the level of support (see text).

this context is that it does not rely on individual hypotheses being accurate: rather it relies on enough hypotheses easily being generatable, and by the same token, discardable.

Finally, we return to the above comment about obtaining improved deletion of points on lines that have already been located. Suppose the cross-section of a line is characterized by a (lateral) Gaussian distribution of edge points. As a true Gaussian extends to infinity in either direction, the support region is not well defined, but for high accuracy it is reasonable to take it as the region within $\pm\sigma$ of the center-line of the line. However, if just these points are eliminated, the remaining points near the line could later on give rise to alternative lines or combine with other points to lead to false positives. It is therefore, better

(Mastorakis and Davies, 2011) to make the "delete distance" d_d larger than the "fit distance" d_f that is used for support during detection, e.g., $d_d = 2\sigma$ or 3σ, where $d_f = \sigma$. ($d_d \approx 3\sigma$ can be considered to be close to optimal because 99.9% of the samples in a Gaussian distribution lie within $\pm 3\sigma$.) Fig. 23.3 shows instances in which the fit distance is 3 pixels and the delete distance has values of 3, 6, 10, and 11 pixels, showing the advantages that can be gained by making d_d significantly greater than d_f. Fig. 23.4 shows a flowchart of the algorithm used in this case.

10.5 LOCATION OF LAPAROSCOPIC TOOLS

The previous section showed how RANSAC can provide a highly efficient means for locating straight edges in digital pictures, and gave an example of its use to locate the handles of laparoscopic tools. These are used for various forms of "key-hole" surgery: specifically, one tool (e.g., a cutter) might be inserted through one incision and another (e.g., a gripper) through a second incision. Additional incisions are needed for viewing via a laparoscope which employs optical fiber technology, and for inflating the cavity—for example, the abdominal or chest cavity. In this section, we consider what information can be obtained via the laparoscope.

Fig. 10.4C shows laparoscopic gripper and cutter tools being located in a simulated flesh background. The latter will normally be a wet surface that is largely red and will exhibit many regions that are close to being specularly reflective. The large variations in intensity that occur under these conditions make the scene quite difficult to interpret. Although the surgeon who is in control of the instruments can learn a lot about the scene through tactile feedback and thus bolster his understanding of it, other people viewing the scene, e.g., on a TV monitor or computer, are liable to find it highly confusing. The same will apply for any computer attempting to interpret, analyze, or record the progress of an operation. These latter tasks are potentially important for logging operations, for training other doctors, for communicating with specialists elsewhere, or for analyzing the progress of the operation during any subsequent debriefing. It would therefore be useful if the exact locations, orientations, and other parameters of the tools could be determined at least relative to the frame of reference of the laparoscope. To this end, RANSAC has provided important 2-D data on the location of the tool handles. Assessing the vanishing points of the pairs of lines from the handles also provides 3-D information. Clearly, further information can be obtained from the coordinates of the ends of the tools.

To identify the ends of the tools, the ends of the handles are first located: this is a straightforward task requiring knowledge of the exact ends of the RANSAC support regions for the tool handle edges. The remainder of the tool ends can now be located by initial approximate prediction, adaptive thresholding, and connected components analysis (Fig. 10.5), special attention being paid to accurate location of the tips of the tool ends. In Fig. 10.5 this is achieved within ~ 1 pixel for the gripper on the left, and slightly less accurately for the closed cutter on the right.

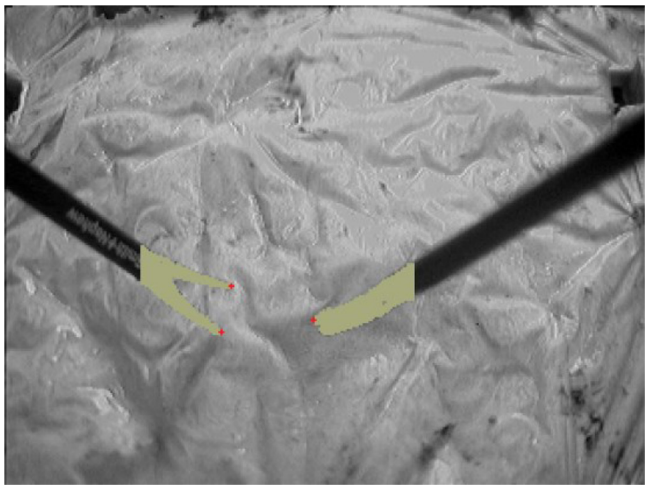


FIGURE 10.5

Tips of laparoscopic tools located from the parts highlighted in gray.

If the gripper had been open, accuracy would have been similar to that for the gripper. Note that, because of the complex intensity patterns in the background, it would have been difficult to locate the tool ends without first identifying the tool handles.

Each of the laparoscopic tools referred to above has (X, Y, Z) position coordinates, together with rotations ψ within the image plane (x, y), θ away from the image plane (toward the Z-axis), and φ about the axis of the handle; in addition, each tool end has an opening angle α (Fig. 10.6). It is bound to be difficult to obtain all seven parameters with any great accuracy from a single monocular view. However, in principle, using an exact CAD model of the tool end, this should be possible with $\sim 15-20^\circ$ accuracy for the angles. The 2-D information about the centerlines and widths of the handles, the convergence of the handle edges, the exact positions of the tips of the tools, should together permit such a 3-D analysis to be carried out. Here we have concentrated on the 2-D analysis: details of the relevant 3-D background theory needed to proceed further can be found in Part 3.

10.6 HOUGH-BASED SCHEMES FOR CIRCULAR OBJECT DETECTION

In this section we present a HT-based approach for circular object detection: the purpose will be to replace the non-robust centroidal profile method outlined in Chapter 9—which is usefully summarized in Table 10.2.

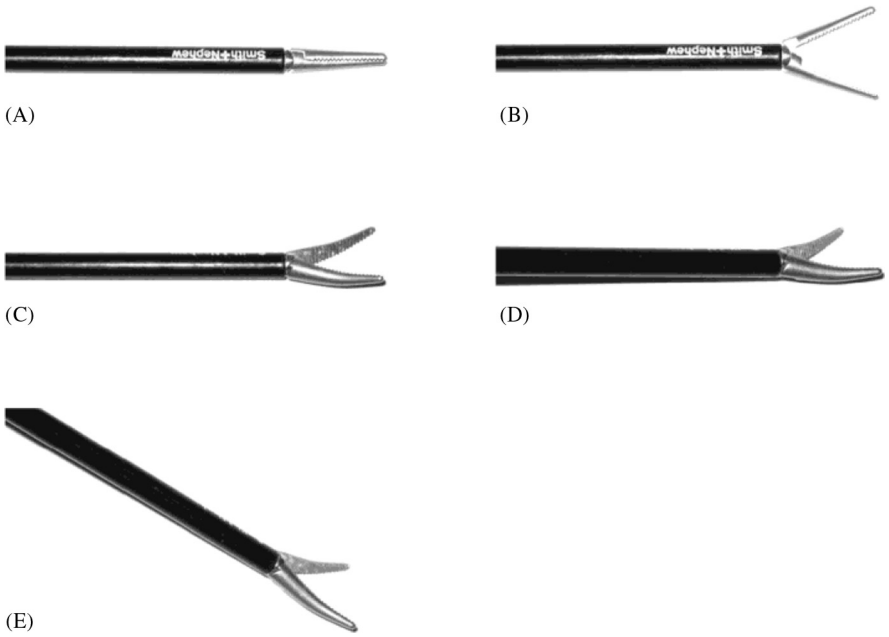


FIGURE 10.6

Orientation parameters for a laparoscopic tool. (A) A gripper tool with closed jaws. (B) Gripper with jaws separated by an angle α. (C) Gripper rotated through an angle φ about a horizontal axis. (D) Gripper tipped through an angle θ away from the image plane. (E) Gripper rotated though an angle ψ about the camera optical axis.

Table 10.2 Procedure for Finding Objects Using (r, θ) Boundary Graphs

1. Locate edges within the image
2. Link broken edges
3. Thin thick edges
4. Track around object outlines
5. Generate a set of (r, θ) plots
6. Match (r, θ) plots to standard templates

This procedure is not sufficiently robust with many types of real data e.g., in the presence of noise, distortions in product shape, etc.: in fact, it is quite common to find the tracking procedure veering off and tracking around shadows or other artifacts.

In the original HT method for finding circles (Duda and Hart, 1972), the intensity gradient is first estimated at all locations in the image and then thresholded to give the positions of significant edges. Then the positions of all possible center locations—namely all points a distance R away from every edge pixel—are accumulated in parameter space, R being the anticipated circle radius. Parameter space can be a general storage area but when looking for circles it is convenient to

make it congruent to image space: in that case possible circle centers are accumulated in a new plane of image space. Finally, parameter space is searched for peaks which correspond to the centers of circular objects. Since edges have nonzero width and noise will always interfere with the process of peak location, accurate center location requires the use of suitable averaging procedures (Davies, 1984c; Brown, 1984).

This approach clearly requires a very large number of points to be accumulated in parameter space and so a revised form of the method has now become standard: in this approach, locally available edge orientation information at each edge pixel is used to enable the exact positions of circle centers to be estimated (Kimme et al., 1975). This is achieved by moving a distance R along the edge normal at each edge location. Thus the number of points accumulated is equal to the number of edge pixels in the image: this represents a significant saving in computational load. (We assume here that objects are known to be *either* lighter *or* darker than the background, so that it is only necessary to move along the edge normal in one direction.) For this procedure to be practicable, the edge detection operator that is employed must be highly accurate. Fortunately, the Sobel operator is able to estimate edge orientation to 1° and is very simple to apply (Chapter 5: Edge Detection). Thus the revised form of the transform is viable in practice.

As was seen in Chapter 5, Edge Detection, once the Sobel convolution masks have been applied, the local components of intensity gradient g_x and g_y are available, and the magnitude and orientation of the local intensity gradient vector can be computed using the formulae:

$$g = (g_x^2 + g_y^2)^{1/2} \tag{10.12}$$

and

$$\theta = \arctan(g_y / g_x) \tag{10.13}$$

However, use of the arctan operation is not necessary when estimating center location coordinates (x_c, y_c) since the trigonometric functions can be made to cancel out:

$$x_c = x - R(g_x / g) \tag{10.14}$$

$$y_c = y - R(g_y / g) \tag{10.15}$$

the values of $\cos\theta$ and $\sin\theta$ being given by:

$$\cos\theta = g_x / g \tag{10.16}$$

$$\sin\theta = g_y / g \tag{10.17}$$

In addition, the usual edge thinning and edge linking operations—which normally require considerable amounts of processing—can be avoided if a little extra smoothing of the cluster of candidate center points is performed (Davies, 1984c) (Table 10.3). Thus this Hough-based approach can be a very efficient one for locating the centers of circular objects, virtually all the superfluous operations having

Table 10.3 A Hough-Based Procedure for Locating Circular Objects

1. Locate edges within the image
2. Link broken edges
3. Thin thick edges
4. For every edge pixel, find a candidate center point
5. Locate all clusters of candidate centers
6. Average each cluster to find accurate center locations

This procedure is particularly robust. It is largely unaffected by shadows, image noise, shape distortions, and product defects. Note that stages 1–3 of the procedure are identical to stages 1–3 in Table 10.2. However, in the Hough-based method, computation can be saved, and accuracy actually increased, by omitting stages 2 and 3.

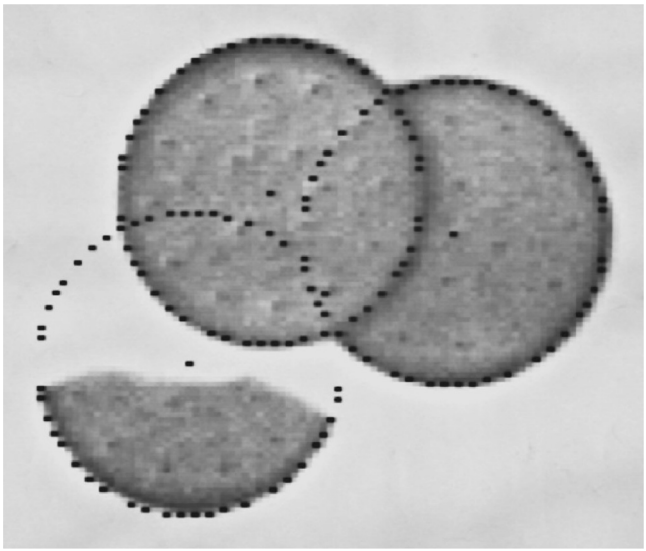


FIGURE 10.7

Location of broken and overlapping biscuits, showing the robustness of the center location technique. Accuracy is indicated by the black dots which are each within 1/2 pixel of the radial distance from the center.

been eliminated, leaving only edge detection, location of candidate center points, and center point averaging to be carried out. In addition, the method is highly robust, so if part of the boundary of an object is obscured or distorted, the object center is still located accurately. In fact, the results are often quite impressive (see e.g., Figs. 10.7 and 10.8). The reason for this useful property is clear from Fig. 10.9.

The efficiency of the above technique means that it takes slightly less time to perform the actual HT part of the calculation than to evaluate and threshold the

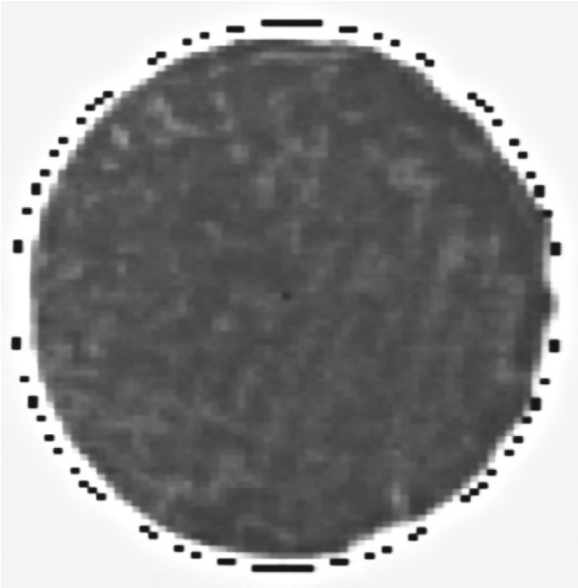


FIGURE 10.8

Location of a biscuit with a distortion, showing a chocolate-coated biscuit with excess chocolate on one edge. Note that the computed center has not been "pulled" sideways by the protuberances. For clarity, the black dots are marked 2 pixels outside the normal radial distance.

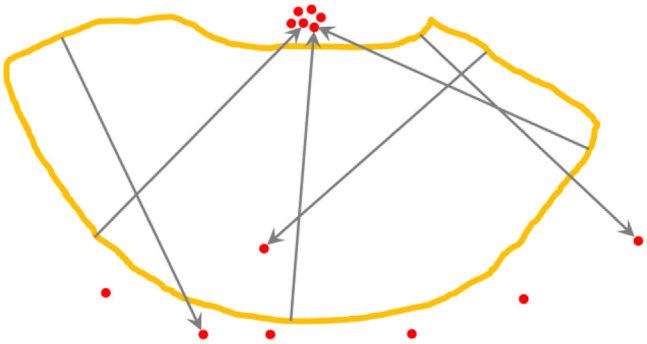


FIGURE 10.9

Robustness of the Hough transform when locating the center of a circular object. The circular part of the boundary gives candidate center points that focus on the true center, whereas the irregular broken boundary gives candidate center points at random positions. In this case, the boundary is approximately that of the broken biscuit shown in Fig. 10.7.

intensity gradient over the whole image. Part of the reason for this is that the edge detector operates within a 3×3 neighborhood and necessitates some 12 pixel accesses, four multiplications, eight additions, two subtractions, and an operation for the evaluation of the square root of sum of squares (Eq. (10.12)).

Overall, the dictates of accuracy imply that candidate center location requires significant computation. However, substantial increases in speed are still possible by software means alone, as will be seen later in the chapter. Meanwhile, we consider the problems that arise when images contain circles of many different radii, or for one reason, or another radii are not known in advance.

10.7 THE PROBLEM OF UNKNOWN CIRCLE RADIUS

There are a number of situations where circle radius is initially unknown. One such situation is where a number of circles of various sizes are being sought—as in the case of coins, or different types of washer. Another is where the circle size is variable—as for food products such as biscuits—so that some tolerance must be built into the system. In general, all circular objects have to be found and their radii measured. In such cases, the standard technique is to accumulate candidate center points simultaneously in a number of parameter planes in a suitably augmented parameter space, each plane corresponding to one possible radius value. The centers of the peaks detected in parameter space give not only the location of each circle in two dimensions but also its radius. Although this scheme is entirely viable in principle, there are several problems in practice:

1. many more points have to be accumulated in parameter space;
2. parameter space requires much more storage;
3. significantly greater computational effort is involved in searching parameter space for peaks.

To some extent this is to be expected, since the augmented scheme enables more objects to be detected directly in the original image.

It is shown below that the last two problems may largely be eliminated. This is achieved by using just one parameter plane to store all the information for locating circles of different radii, i.e., accumulating not just one point per edge pixel but a whole line of points along the direction of the edge normal in this one plane. In practice, the line need not be extended indefinitely in either direction but only over the restricted range of radii over which circular objects or holes might be expected.

Even with this restriction, a large number of points are being accumulated in a single parameter plane, and it might be thought initially that this would lead to such a proliferation of points that almost any "blob" shape would lead to a peak in parameter space which might be interpreted as a circle center. However, this is not so and significant peaks normally result only from genuine circles and closely related shapes.

To understand the situation, consider how a sizeable peak can arise at a particular position in parameter space. This can happen only when a large number of

radial vectors from this position meet the boundary of the object normally. In the absence of discontinuities in the boundary, a contiguous set of boundary points can only be normal to radius vectors if they lie on the arc of a circle. Nevertheless, errors in the measurement of local edge orientation make the scheme slightly less specific in its capability for detecting circular objects.

Finally, note that the information on radial distance has been lost by accumulating all votes in a single parameter plane. Hence, a further stage of analysis is needed to measure object radius. This extra stage of analysis normally involves negligible additional computation, because the search space has been narrowed down so severely by the initial circle location procedure: hence only a 1-D HT need be used, with radial distance as the relevant parameter.

10.7.1 PRACTICAL RESULTS

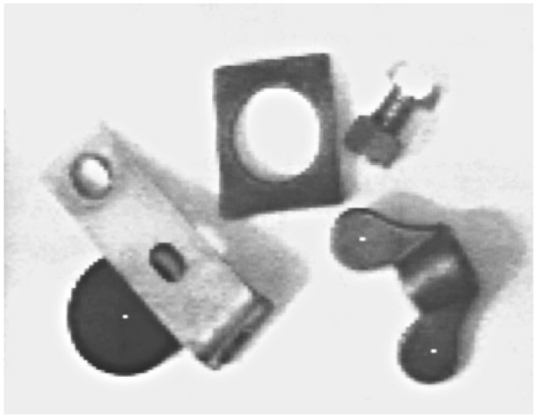
The method described above works much as expected, the main problems arising with small circular objects (of radii less than about 20 pixels) at low resolution (Davies, 1988b). Essentially, the problems are those of lack of discrimination in the precise shapes of small objects (see Fig. 10.10), as anticipated above. As suggested earlier, this can frequently be turned to advantage in that the method becomes a circular feature detector for small radii (see Fig. 10.10, where a wing nut is located).

As required, objects are detected reliably even when they are partly occluded. However, it is clear from Fig. 10.10 that high accuracy of center location cannot be expected when a single parameter plane is used to detect objects over a large range of sizes: hence it is best to cut down the voting range as far as possible.

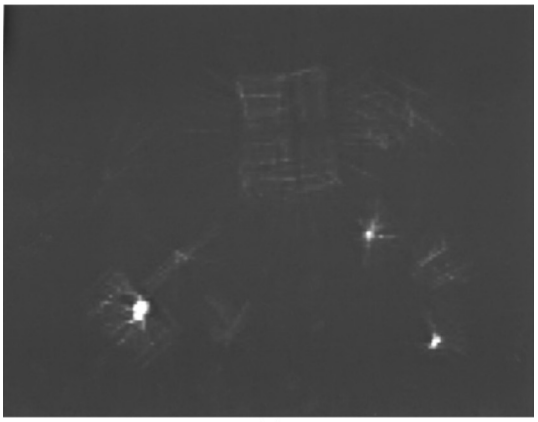
Overall, there is a tradeoff between speed and accuracy with this approach. However, the results confirm that it is possible to locate objects of various radii within a significantly conflated parameter space, thereby making substantial savings in storage and computation—even though the total number of votes that have to be accumulated in parameter space is not itself reduced.

10.8 OVERCOMING THE SPEED PROBLEM

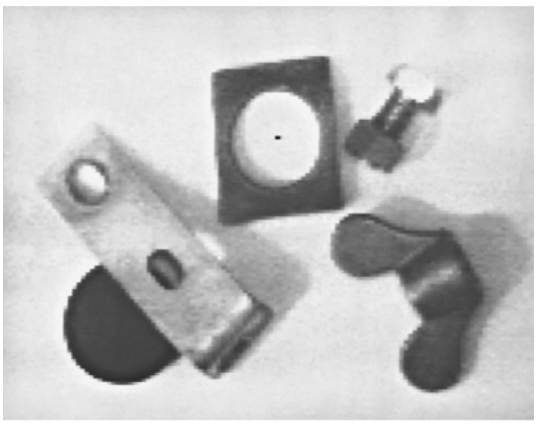
This section examines how circle detection may be carried out with significant improvement in speed. To achieve this, two methods are tried: (1) sampling the image data and (2) using a simpler edge detector. The most appropriate strategy for (1) appears to be to look only at every nth line in the image, while that for (2) involves using a small 2-element neighborhood while searching for edges (Davies, 1987d). Although this approach will lose the capability for estimating edge orientation, it will still permit horizontal and vertical chords of a circle to be bisected, thereby leading to values for the center coordinates x_c, y_c. It also involves much less computation, the multiplications and square root calculations, and most of the divisions being eliminated or replaced by 2-element differencing



(A)



(B)



(C)

FIGURE 10.10

(A) Accurate simultaneous detection of a lens cap and a wing nut when radii are assumed to range from 4 to 17 pixels; (B) response in parameter space that arises with such a range of radii: note the overlap of the transforms from the lens cap and the bracket; (C) hole detection in the image of (A) when radii are assumed to fall in the range −26 to −9 pixels (negative radii are used since holes are taken to be objects of negative contrast): clearly, in *this* image a smaller range of negative radii could have been employed.

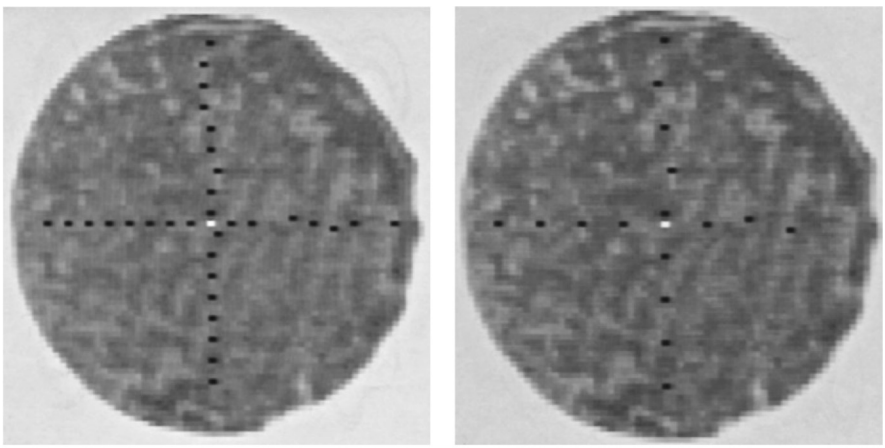
operations. Practical details showing the overall speed gain available using this approach are given below.

Both the original HT and the chord bisection approach lead to peak formation, though the former leads to a single 2-D peak and the latter to two 1-D peaks which have to be obtained separately. Robustness of circle detection depends on all peaks being found reliably, and this becomes less likely when objects are distorted. Clearly, if a reduced number of horizontal and vertical scan lines contribute to the 1-D peaks, this will also cause a potential reduction in the robustness of detection, i.e., the risk that a peak will be missed. In addition, there is a further factor to be considered: if only a proportion α of all possible horizontal and vertical scan lines contribute to 1-D HT peaks, the signal-to-noise ratio will fall by a factor $\sqrt{\alpha}$ and the accuracy with which the center can be located will be similarly reduced.

While the chord sampling strategy can be highly effective, it is susceptible to problems with highly textured objects: this is because too many false edges may be produced, and these can lead to chords that do not stretch across the whole object; as this leads to a further lowering of α and further reduces the robustness and accuracy of the method.

10.8.1 PRACTICAL RESULTS

Tests (Davies, 1987d) with the images in Fig. 10.11 showed that gains in speed of more than 25 can be obtained, with values of α down to less than 0.1



(A) (B)

FIGURE 10.11

Successful object location using the chord bisection algorithm for the same initial image, using successive step sizes of 4 and 8 pixels. The black dots show the positions of the horizontal and vertical chord bisectors, and the white dot shows the position found for the center.

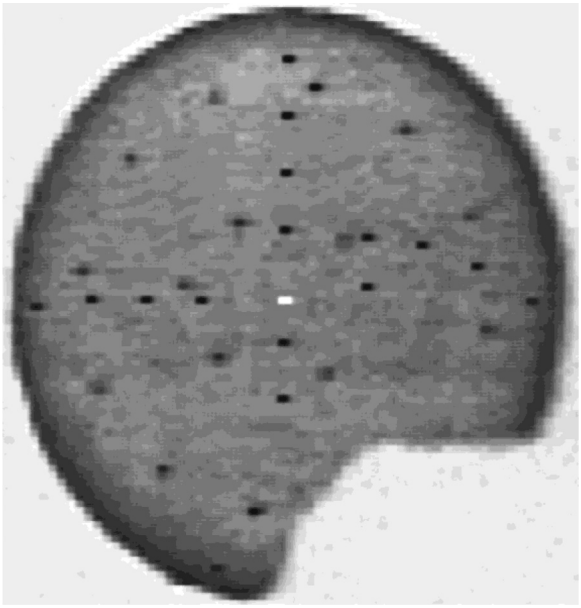


FIGURE 10.12

Successful location of a broken object using the chord bisection algorithm: only about one-quarter of the ideal boundary is missing.

(i.e., every 10th horizontal and vertical line scanned). The results for broken circular products (Fig. 10.12) are self-explanatory; they indicate the limits to which the method can be taken. An outline of the complete algorithm is given in Table 10.4 (note the relatively straightforward problem of disambiguating the results if there happen to be several peaks).

Fig. 11.13 shows the effect of adjusting the threshold in the 2-element edge detector. The result of setting it too low is seen in Fig. 10.13A. Here the surface texture of the object has triggered the edge detector, and the chord midpoints give rise to a series of false estimates of the center coordinates. Fig. 11.13B shows the result of setting the threshold at too high a level, so that the number of estimates of center coordinates is reduced and sensitivity suffers.

Overall, the center location procedure described above is more than an order of magnitude faster than the standard HT and often as much as 25 times faster. This could be quite important in exacting real-time applications. Robustness is so good that the method tolerates at least one-quarter of the circumference of an object being absent, making it adequate for many real applications. Importantly, it is entirely clear what types of image data would be likely to confuse the algorithm.

Table 10.4 Outline of the Fast Center-Finding Algorithm

```
y = 0;
do {
    scan horizontal line y looking for start and end of each object;
    calculate midpoints of horizontal object segments;
    accumulate midpoints in 1-D parameter space (x space);
    // note that the same space, x space, is used for all lines y
    y = y + d;
} until y > ymax;
x = 0;
do {
    scan vertical line x looking for start and end of each object;
    calculate midpoints of vertical object segments;
    accumulate midpoints in 1-D parameter space (y space);
    // note that the same space, y space, is used for all lines x
    x = x + d;
} until x > xmax;

find peaks in x space;
find peaks in y space;
test all possible object centres arising from these peaks;
// the last step is necessary only if ∃ > 1peak in each space
// d is the horizontal and vertical step-size (= 1/α)
```

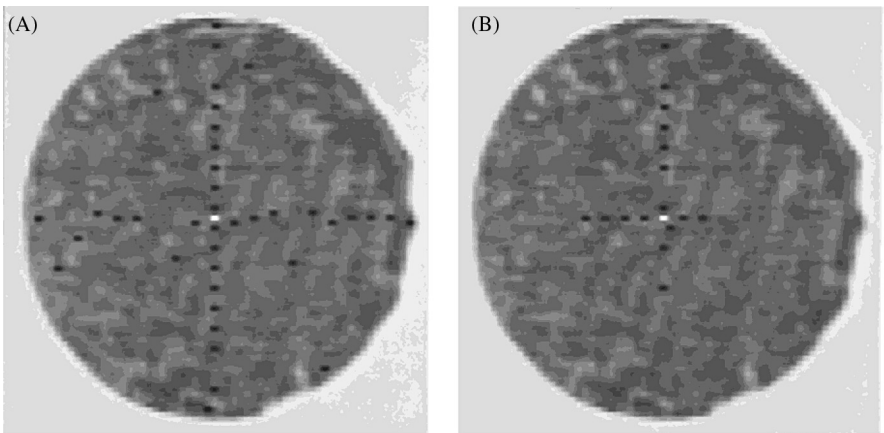


FIGURE 10.13

Effect of misadjustment of the gradient threshold: (A) effect of setting the threshold too low, so that surface texture muddles the algorithm; (B) loss of sensitivity on setting the threshold too high.

10.9 ELLIPSE DETECTION

The problem of detecting ellipses may seem only marginally more complex than that of detecting circles—as eccentricity is only a single parameter. However, eccentricity destroys the symmetry of the circle, so the direction of the major axis also has to be defined. As a result, five rather than four parameters are required to describe an ellipse, and ellipse detection has to take account of this, either explicitly or implicitly. In spite of this, one method for ellipse detection is especially simple and straightforward to implement: that is the diameter bisection method, which is described next.

10.9.1 THE DIAMETER BISECTION METHOD

The diameter bisection method of Tsuji and Matsumoto (1978) is very simple in concept. First, a list is compiled of all the edge points in the image. Then, the list is sorted to find those that are antiparallel, so that they could lie at opposite ends of ellipse diameters; next, the positions of the center points of the connecting lines for all such pairs are taken as voting positions in parameter space (Fig. 10.14). As for circle location, the parameter space that is used for this purpose is congruent to image space. Finally, the positions of significant peaks in parameter space are located to identify possible ellipse centers.

Naturally, in an image containing many ellipses and other shapes, there will be very many pairs of antiparallel edge points and for most of these the center points of the connecting lines will lead to nonuseful votes in parameter space. Clearly,

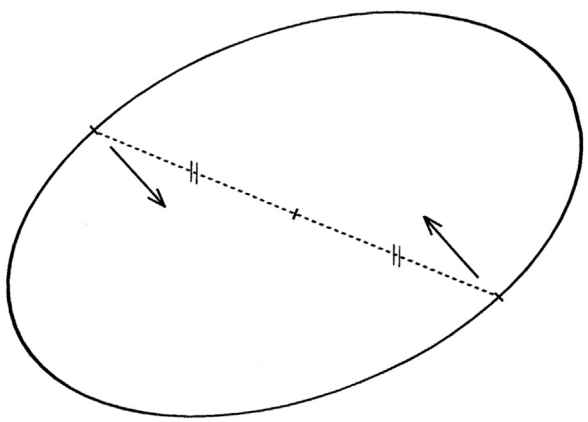


FIGURE 10.14

Principle of the diameter bisection method. A pair of points is located for which the edge orientations are antiparallel. If such a pair of points lies on an ellipse, the midpoint of the line joining the points will be situated at the center of the ellipse.

such clutter leads to wasted computation. However, it is a principle of the HT that votes must be accumulated in parameter space at all points which *could in principle* lead to correct object center location: it is left to the peak finder to find the voting positions that are most likely to correspond to object centers.

Not only does clutter lead to wasted computation but the method itself is computationally expensive. This is because it examines all *pairs* of edge points, and there are many more such pairs than there are edge points (m edge points lead to ${}^{m}C_{2} \approx m^{2}/2$ pairs of edge points). Indeed, since there are likely to be at least 1000 edge points in a typical image, the computational problems can be formidable.

Interestingly, the basic method is not particularly discriminating about ellipses. It picks out many symmetrical shapes—any indeed that possess 180° rotation symmetry, including rectangles, ellipses, circles, or superellipses (these have equations of the form $x^{s}/a^{s} + y^{s}/b^{s} = 1$, ellipses being a special case). In addition, the basic scheme sometimes gives rise to a number of false identifications even in an image in which only ellipses are present (Fig. 10.15). However, Tsuji and Matsumoto (1978) also proposed a technique by which true ellipses can be distinguished. The basis of the technique is the property of an ellipse that the lengths of perpendicular semidiameters OP, OQ (O being the centre of the ellipse and P and Q being boundary points) obey the relation:

$$1/\mathrm{OP}^{2} + 1/\mathrm{OQ}^{2} = 1/R^{2} = \text{constant} \tag{10.18}$$

To proceed, the set of edge points that contribute to a given peak in parameter space is used to construct a histogram of R values (the latter being obtained from Eq. (10.18)). If a significant peak is found in this histogram, then there is clear

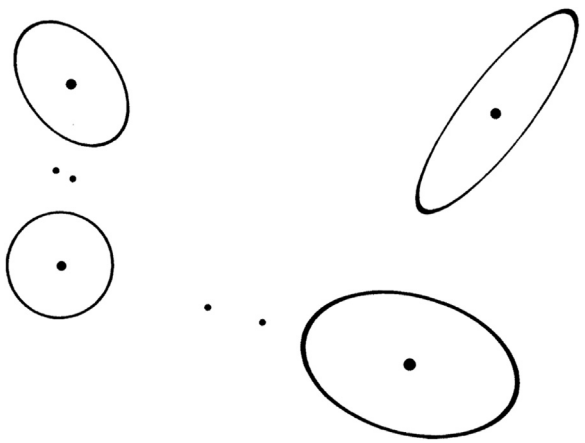


FIGURE 10.15

Result of using the basic diameter bisection method. The large dots show true ellipse centers found by the method, while the smaller dots show positions at which false alarms commonly occur. Such false alarms are eliminated by applying the test described in the text.

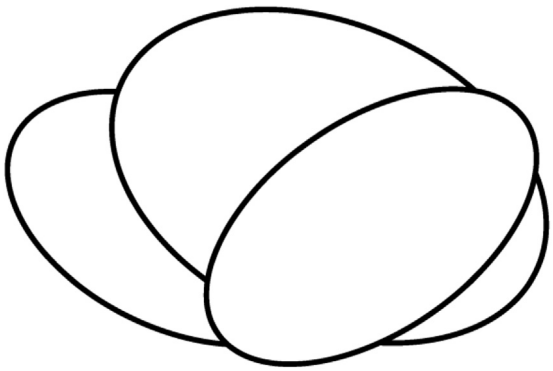


FIGURE 10.16

Limitations of the diameter bisection method: of the three ellipses shown, only the leftmost one cannot be located by the diameter bisection method.

evidence of an ellipse at the specified location in the image. If two or more such peaks are found, then there is evidence of a corresponding number of concentric ellipses in the image. If, however, no such peaks are found, then a rectangle, superellipse, or other symmetrical shape may be present and each of these would need its own identifying test.

The method obviously relies on there being an appreciable number of pairs of edge points on an ellipse lying at opposite ends of diameters: hence there are strict limits on the amount of the boundary that must be visible (Fig. 10.16). Finally, it should not go unnoticed that the method wastes the signal available from unmatched edge points. These considerations have led to a search for further methods of ellipse detection.

10.9.2 THE CHORD–TANGENT METHOD

The chord–tangent method was devised by Yuen et al. (1988) and makes use of another simple geometric property of the ellipse. Again pairs of edge points are taken in turn, and for each point of the pair, tangents to the ellipse are constructed and found to cross at T; the midpoint of the connecting line is found at M; then the equation of line TM is calculated and all points that lie on the portion MK of this line are accumulated in parameter space (Fig. 10.17) (clearly, T and the center of the ellipse lie on the opposite sides of M). Finally, peak location proceeds as before.

The proof that this method is correct is trivial. Symmetry ensures that the method works for circles, and projective properties then ensure that it also works for ellipses: under orthographic projection (see Chapter 16: The Three-Dimensional World), straight lines project into straight lines, midpoints into midpoints, tangents into tangents, and circles into ellipses; in addition, it is always

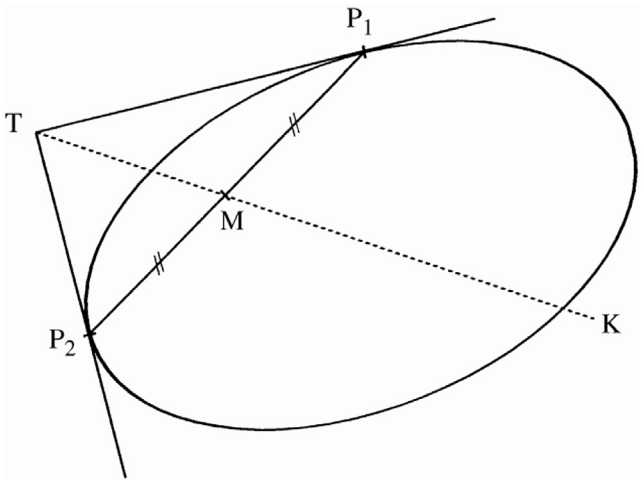


FIGURE 10.17

Principle of the chord–tangent method. The tangents at P_1 and P_2 meet at T and the midpoint of P_1P_2 is M. The center C of the ellipse lies on the line TM produced. Notice that M lies between C and T. Hence the transform for points P_1 and P_2 need only include the portion MK of this line.

possible to find a viewpoint such that a circle can be projected into a given ellipse.

Unfortunately, this method suffers from significantly increased computation, since so many points have to be accumulated in parameter space. This is obviously the price to be paid for greater applicability. However, computation can be minimized in at least three ways: (1) cutting down the lengths of the lines of votes accumulated in parameter space by taking account of the expected sizes and spacings of ellipses; (2) not pairing edge points initially if they are too close together or too far apart; and (3) eliminating edge points once they have been identified as belonging to a particular ellipse.

10.9.3 FINDING THE REMAINING ELLIPSE PARAMETERS

Although the methods described above are designed to locate the center coordinates of ellipses, a more formal approach is required to determine other ellipse parameters. Accordingly, we write the equation of an ellipse in the form:

$$Ax^2 + 2Hxy + By^2 + 2Gx + 2Fy + C = 0 \tag{10.19}$$

an ellipse being distinguished from a hyperbola by the additional condition:

$$AB > H^2 \tag{10.20}$$

This condition guarantees that A can never be zero and that the ellipse equation may without loss of generality be rewritten with $A = 1$. This leaves five parameters, which can be related to the position of the ellipse, its orientation, and its size and shape (or eccentricity).

Having located the center of the ellipse, we may select a new origin of coordinates at its center (x_c, y_c); the equation then takes the form:

$$x'^2 + 2Hx'y' + By'^2 + C' = 0 \tag{10.21}$$

where

$$x' = x - x_c; \quad y' = y - y_c \tag{10.22}$$

It now remains to fit to Eq. (10.21) the edge points that gave evidence for the ellipse center under consideration. The problem will normally be vastly overdetermined. Hence an obvious approach is the method of least squares. Unfortunately, this technique tends to be very sensitive to outlier points and is therefore liable to be inaccurate. An alternative is to employ some form of HT. Here, we follow Tsukune and Goto (1983) by differentiating Eq. (10.21):

$$x' + By'/dx' + H(y' + x'\mathrm{d}y'/\mathrm{d}x') = 0 \tag{10.23}$$

Then $\mathrm{d}y'/\mathrm{d}x'$ can be determined from the local edge orientation at (x', y') and a set of points accumulated in the new (H, B) parameter space. When a peak is eventually located in (H, B) space, the relevant data (a subset of a subset of the original set of edge points) can be used with Eq. (10.21) to obtain a histogram of C' values, from which the final parameter for the ellipse can be obtained.

The following formulae are needed to determine the orientation θ and semiaxes a, b of an ellipse in terms of H, B, and C':

$$\theta = \frac{1}{2}\arctan\left(\frac{2H}{1-B}\right) \tag{10.24}$$

$$a^2 = \frac{-2C'}{(B+1) - [(B-1)^2 + 4H^2]^{1/2}} \tag{10.25}$$

$$b^2 = \frac{-2C'}{(B+1) + [(B-1)^2 + 4H^2]^{1/2}} \tag{10.26}$$

Mathematically, θ is the angle of rotation that diagonalizes the second-order terms in Eq. (10.21); having performed this diagonalization, the ellipse is then essentially in the standard form $\tilde{x}^2/a^2 + \tilde{y}^2/b^2 = 1$, so a and b are determined.

Note that the above method finds the five ellipse parameters in *three* stages: first the positional coordinates are obtained, then the orientation, and finally the size and eccentricity. (Strictly, the eccentricity is $e = (1 - b^2/a^2)^{1/2}$, but in most cases we are more interested in the ratio of semiminor to semimajor axes, b/a.) This three-stage calculation involves less computation but compounds any errors—in addition, edge orientation errors, though low, become a limiting factor. For this reason, Yuen et al. (1988) tackled the problem by speeding up the HT

procedure itself rather than by avoiding a direct assault on Eq. (10.21): i.e., they aimed at a fast implementation of a thoroughgoing second stage which finds all the parameters of Eq. (10.21) in one 3-D parameter space.

It is now clear that reasonably optimal means are available for finding the orientation and semiaxes of an ellipse once its position is known: the weak point in the process appears to be that of finding the ellipse initially. Indeed, the two approaches for achieving this that have been described above are particularly computation intensive, mainly because they examine all pairs of edge points; a possible alternative is to apply the generalized Hough transform (GHT), which locates objects by taking edge points singly: this possibility will be considered in Chapter 11, The Generalized Hough Transform.

10.10 HUMAN IRIS LOCATION

Human iris location is an important application of computer vision for three reasons: (1) it provides a useful cue for human face analysis; (2) it can be used for the determination of gaze direction; (3) it is useful in its own right for biometric purposes, that is to say, for identifying individuals almost uniquely. The latter possibility has already been noted in Chapter 7, Texture Analysis where textural methods for iris recognition were outlined and some key references were given. More details of human face location and analysis will be given in Chapter 21, Face Detection and Recognition: the Impact of Deep Learning. Here we concentrate on iris location using the HT.

In fact, we can tackle the iris location and recognition task reasonably straightforwardly. First, if the head has been located with reasonable accuracy, then it can form a region of interest, inside which the iris can be sought. In a front view with the eyes looking ahead, the iris will be seen as a round often high-contrast object, and can be located straightforwardly with the aid of a HT (Ma et al., 2003). In some cases this will be less easy because the iris is relatively light and the color may not be distinctive—though a lot will depend on the quality of the illumination. Perhaps more important, in some human subjects, the eyelid and lower contour of the eye may partially overlap the iris (Fig. 10.18A), making it more difficult to identify, though, as confirmed below, HTs are capable of coping with a fair degree of occlusion.

Note that the iris will appear elliptical if the eyes are not be facing directly ahead; in addition, the shape of the eye is far from spherical and the horizontal diameter is larger than the vertical diameter—again making the iris appear elliptical (Wang and Sung, 2001). In either case the iris can still be detected using a HT. Furthermore, once this has been done, it should be possible to estimate the direction of gaze with a reasonable degree of accuracy (Gong et al., 2000), thereby taking us further than mere recognition. (The fact that measurement of ellipse eccentricity would lead to an ambiguity in the gaze direction can be offset by measuring the position of the ellipse on the eyeball.) Finally, Toennies et al. (2002) showed that the HT can be used to localize the irises for real-time

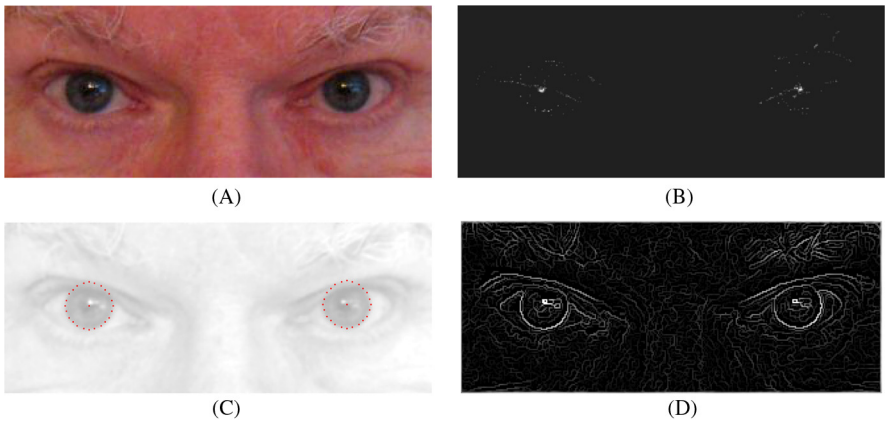


FIGURE 10.18

Iris location using the Hough transform. (A) Original image of the eye region of a face. (B) The gradient-weighted Hough transform (HT) in parameter space. (c) Accurate location of the irises from the peaks in (B). (D) Output from the Canny operator (incorporating smoothing, nonmaximum suppression and hysteresis) used to obtain the initial edge image. Gradient weighting has contributed significantly to the robustness and accuracy of object (iris) location. Notice the plethora of additional edges in (D): these give rise to substantial numbers of votes which interfere with those from the irises.

applications, in spite of quite substantial partial occlusion by the eyelid and lower contour of the eye.

A number of the points made above are illustrated by the example in Fig. 10.18. Far from being a trivial application of the HT, there are a surprisingly large number of edges in the eye region: these produce significant numbers of additional votes which interfere with those from the irises. These arise from the upper and lower eyelids and the folds of skin around them. Hence the accuracy of the method is not assured: this makes gradient weighting (see Section 11.4) especially valuable. The radii of the irises shown in Fig. 10.18 are about 17.5 pixels, and there is no particular evidence of ellipticity in their shape. However, more accurate location of the iris and measurement of ellipticity in order to estimate orientation (e.g., to determine the angle of gaze) require considerably greater resolution, with iris radii approaching 100 pixels, whereas pictures that analyze iris texture patterns for biometric purposes require even larger radii.

10.11 CONCLUDING REMARKS

This chapter has described a variety of techniques for finding straight lines and straight edges in digital images. Several of these were based on the HT, which is

important because it permits systematic extraction of global data from images and has the capability to ignore "local" problems, due for example, to occlusions and noise. This is what is required for "intermediate-level" processing, as will be seen repeatedly in later chapters.

The specific techniques covered have involved various parametrizations of a straight line, and means for improving efficiency and accuracy. In particular, speed is improved by using a two-stage line finding procedure—a method that is useful in other applications of the HT, as will be seen in later chapters. Accuracy tends to be reduced by such two-stage processing because of error propagation and because the first stage is liable to be subject to too many interfering signals. However, it is possible to improve the accuracy of approximate solutions by using least squares refinement procedures.

Subsequently, it became clear that the RANSAC approach also has line fitting capabilities, which are for some purposes superior to those of the HT, though RANSAC tends to be more computation intensive (with N edge points it has a computational load of $O(N^3)$ rather than $O(N^2)$). Suffice it to say that the final choice of approach will depend on the exact type of image data, including levels of noise and background clutter.

The chapter went on to describe techniques for circle and ellipse detection, starting with the HT approach. Although the HT is found to be effective and highly robust against occlusions, noise, and other artifacts, it requires considerable storage and computation—especially if circles of unknown radius are to be located. A method has been described for efficiently tackling the latter problem using a single 2-D parameter space. In addition, a technique has been described for markedly reducing the computational load involved in circle detection by sampling along every nth row and column. In essence, the technique replaces a 2-D search by two 1-D searches, which though more efficient limits the robustness and accuracy in a known way. This is in line with the principle that (as for the HT), robustness cannot be added as an afterthought but must be included as an integral part of the design of any vision algorithm.

Two HT-based schemes for ellipse detection have also been described—the diameter bisection method and the chord−tangent method. A further approach to ellipse detection, based on the generalized HT, will be covered in Chapter 11, The Generalized Hough Transform. At that point further lessons will be drawn on the efficacies of the various methods.

As in the case of line detection, a trend running through the design of circle and ellipse detection schemes is the deliberate splitting of algorithms into two or more stages. This is useful for keying into the important and relevant parts of an image prior to finely discriminating one type of object or feature from another, or prior to measuring dimensions or other characteristics accurately. Indeed, the concept can be taken further, in that the efficiencies of all the algorithms discussed in this chapter have been improved by searching first for edge features in the image. The concept of two-stage template matching is therefore deep-seated in the methodology of the subject and is developed further in later chapters.

Although two-stage template matching is a standard means of increasing efficiency (VanderBrug and Rosenfeld, 1977; Davies, 1988f), it is not obvious that efficiency can always be increased in this way. It appears to be in the nature of the subject that ingenuity is needed to discover means of achieving this.

The HT is one way of inferring the presence of objects from their feature points, and RANSAC is another. Both methods use voting schemes to select best fit lines, though only the HT employs a parameter space representation; and both methods are highly robust as they focus only on positive evidence for the existence of objects. The HT also achieves an impressive level of robustness for circle and ellipse detection. Practical issues such as speed and storage requirements can in some cases be improved by employing parameter spaces of reduced dimension.

10.12 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The HT was developed in 1962 (Hough, 1962) with the aim of finding (straight) particle tracks in high-energy nuclear physics, and was brought into the mainstream image analysis literature much later by Rosenfeld (1969). Duda and Hart (1972) developed the method further and applied it to the detection of lines and curves in digital pictures. O'Gorman and Clowes (1976) soon developed a Hough-based scheme for finding lines efficiently, by making use of edge orientation information, at much the same time that Kimme et al. (1975) applied the same method (apparently independently) to the efficient location of circles. Many of the ideas for fast effective line finding described in this chapter arose in a paper by Dudani and Luk (1978). The author's foot-of-normal method (Davies, 1986) was developed much later. During the 1990s, work in this area progressed further—see for example, Atiquzzaman and Akhtar's (1994) method for the efficient determination of lines together with their end coordinates and lengths; Lutton et al.'s (1994) application of the transform to the determination of vanishing points; and Kamat-Sadekar and Ganesan's (1998) extensions of the technique to cover the reliable detection of multiple line segments, particularly in relation to road scene analysis.

Some mention should be made of the related Radon transform. This is formed by integrating the picture function $I(x, y)$ along infinitely thin straight strips of the image, with normal coordinate parameters (θ, ρ), and recording the results in a (θ, ρ) parameter space. The Radon transform is a generalization of the HT for line detection (Deans, 1981). In fact, for straight lines the Radon transform reduces to the Duda and Hart (1972) form of the HT. The transforms of real lines have a characteristic "butterfly" shape (a pencil of segments passing through the corresponding peak) in parameter space. This phenomenon was investigated by Leavers and Boyce (1987), who devised special 3×3 convolution filters for sensitively detecting these peaks.

There has been strong continuing interest in the HT in spite of its computational difficulties: in fact, these reflect underlying matching problems that are inescapable in computer vision, so development of methods must continue. Thus Schaffalitsky and Zisserman (2000) carried out an interesting extension of earlier ideas on vanishing lines and points by considering the case of repeated lines such as those occurring on certain types of fences and brick buildings; Song et al. (2002) developed HT methods for coping with the problems of fuzzy edges and noise in large-sized images; and Guru et al. (2004) demonstrated viable alternatives to the HT, based for example, on heuristic search achieved by small eigenvalue analysis.

The author's work on circle detection for automated inspection required real-time implementation and also high accuracy. This spurred the development of the techniques described in Sections 10.7–10.8 (Davies, 1987d, 1988b). In addition, the author considered the effect of noise on edge orientation computations, showing in particular their effect in reducing the accuracy of center location (Davies, 1987c): see Section 5.9.

Yuen et al. (1989) reviewed various existing methods for circle detection using the HT. In general, their results confirmed the efficiency of the method of Section 10.7 for unknown circle radius, although they found that the two-stage process that was involved can sometimes lead to slight loss of robustness. It appears that this problem can be reduced in some cases by using a modified version of the algorithm of Gerig and Klein (1986); but note that the Gerig and Klein approach is itself a two-stage procedure. More recently, Pan et al. (1995) have increased the speed of computation of the HT by prior grouping of edge pixels into arcs, for an underground pipe inspection application.

The two-stage template matching technique and related approaches for increasing search efficiency in digital images were known by 1977 (Nagel and Rosenfeld, 1972; Rosenfeld and VanderBrug, 1977; VanderBrug and Rosenfeld, 1977), and have undergone further development since then—especially in relation to particular applications such as those described in this chapter (Davies, 1988f).

The ellipse detection sections are based particularly on the work of Tsuji and Matsumoto (1978), Tsukune and Goto (1983), and Yuen et al. (1988); for a fourth method (Davies, 1989a) using the GHT idea of Ballard (1981) in order to save computation, see Chapter 11, The Generalized Hough Transform. The contrasts between these methods are many and intricate, as this chapter has shown. In particular, the idea of saving dimensionality in the implementation of the GHT appears also in a general circle detector (Davies, 1988b). At that point in time, the necessity for a multistage approach to determination of ellipse parameters seemed proven, although somewhat surprisingly the optimum number of such stages was just two.

Later algorithms represented moves to greater degrees of robustness with real data by explicit inclusion of errors and error propagation (Ellis et al., 1992); increased attention was subsequently given to the verification stage of the Hough approach (Ser and Siu, 1995). In addition, work was carried out on the detection

of superellipses, which are shapes intermediate in shape between ellipses and rectangles, though the technique used (Rosin and West, 1995) was that of segmentation trees rather than HTs (nonspecific detection of superellipses can of course be achieved by the diameter bisection method—see Section 10.9.1); see also Rosin (2000).

For cereal grain inspection, with typical flow rates in excess of 300 grains per second, ultra-fast algorithms were needed and the resulting algorithms were limiting cases of chord-based versions of the HT (Davies, 1999a,b); a related approach was adopted by Xie and Ji (2002) for their efficient ellipse detection method; Lei and Wong (1999) employed a method which was based on symmetry, and this was found to be able to detect parabolas and hyperbolas as well as ellipses. Note that while this is advantageous in some applications, the lack of discrimination could prove to be a disadvantage in other applications. It was also reported as being more stable than other methods since it does not have to calculate tangents or curvatures; the latter advantage has also been reported by Sewisy and Leberl (2001). The fact that even in the 2000's, basic new ellipse detection schemes are being developed says something about the science of image analysis: even today the toolbox of algorithms is incomplete, and the science of how to choose between items in the toolbox, or how, *systematically*, to develop new items for the toolbox, is immature. Further, although all the parameters for specification of such a toolbox may be known, knowledge about the possible tradeoffs between them is still limited.

10.12.1 MORE RECENT DEVELOPMENTS

Advances are still being made in the application of the HT. In particular, Chung et al. (2010) have developed an orientation-based elimination strategy that they have shown to be more efficient than previous line-determination methods based on the HT. It operates by dividing edge pixels into sets with small (typically 10°) ranges of orientation, and for each of these, it carries out the process of line detection. Since this process involves a parameter space of reduced size, both storage and search times are reduced.

The RANSAC procedure was published by Fischler and Bolles in 1981: this must be one of the most cited papers in computer vision, and the method must be one of the most used (more even than the HT, because it only relies on the existence of suitable hypotheses, *however* obtained). The original paper used it for tackling the full perspective *n*-point fitting problem in 3-D (see Chapter 17: Tackling the Perspective *n*-Point Problem). Clarke et al. (1996) used it for locating and tracking straight lines. Borkar et al. (2009) used it for locating lane markings on roads, and Mastorakis and Davies (2011) developed it further for the same purpose. Interestingly, Borkar et al. used a low resolution HT to feed RANSAC, and followed it by least squares fitting of the inliers. The paper does not report on how much was gained by this three-stage approach, either in accuracy or in reliability. (If enough hypotheses are employed—and there is certainly

no lack of these in such an application—both the HT and least squares fitting might be avoided, but here optimization for speed may make the inclusion of least squares essential.) For further discussion of RANSAC, see Chapter 23, In-Vehicle Vision Systems and Appendix A.

Much work has recently been carried out on iris detection using the HT. Jang et al. (2008) were particularly concerned with overlap of the iris region by the upper and lower eyelids, and used a parabolic version of the HT to accurately locate their boundaries, taking special care to limit the computational load. Li et al. (2010) used a circular HT to locate the iris and a RANSAC-like technique for locating the upper and lower eyelids, again using a parabolic model for the latter: their approach was designed to cope with very noisy iris images. Chen et al. (2010) used a circular HT to locate the iris and a straight line HT to locate up to two line segment approximations to the boundaries of each of the eyelids. Cauchie et al. (2008) produced a new version of the HT to accurately locate common circle centers from circular or partial circle segments, and demonstrated its value for iris location. Min and Park (2009) used a circular HT for iris detection, a parabolic HT for eyelid detection, and followed this with eyelash detection using thresholding.

Finally, we summarize work carried out by Guo et al. (2009) to overcome the problems of dense sets of edges in textured regions. To reduce the impact of such edges, a measure of isotropic surround suppression was introduced: the resulting algorithm gave small weights to edges in texture regions and large weights to edges on strong and clear boundaries when accumulating votes in Hough space. The approach gave good results when locating straight lines in scenes containing man-made structures such as buildings.

10.13 PROBLEMS

1. **a.** In the foot-of-normal HT, straight edges are found by locating the foot of the normal F (x_f, y_f) from an origin O (0, 0) at the center of the image to an extended line containing each edge fragment E (x, y), and placing a vote at F in a separate image space.
   b. By examining the possible positions of lines within a square image and the resulting foot-of-normal positions, determine the exact extent of the parameter space that is theoretically required to form the HT.
   c. Would this form of the HT be expected to be (1) more or less robust and (2) more or less computation intensive than the (ρ, θ) HT for line location?
2. **a.** Why is it sometimes stated that a HT generates *hypotheses* rather than actual solutions for object location? Is this statement justified?
   b. A new type of HT is to be devised for detecting straight lines. It will take every edge fragment in the image and extend it in either direction until it meets the boundary of the image, and then accumulate a vote at each position. Thus *two* peaks should result for every line. Explain why

finding these peaks will require *less* computation than for the standard HT, but that deducing the presence of lines will then require *extra* computation. How will these amounts of computation be likely to vary with (1) the size of the image and (2) the number of lines in the image?

c. Discuss whether this approach would be an improvement on the standard approach for straight line location, and whether it would have any disadvantages.

3. a. Describe the HT approach to object location. Explain its advantages relative to the *centroidal* (r, θ) *plot* approach: illustrate your answer with reference to location of circles of known radius R.

b. Describe how the HT may be used to locate straight edges. Explain what is seen in the parameter space if many curved edges also appear in the original image.

c. Explain what happens if the image contains a square object of arbitrary size and *nothing* else. How would you deduce from the information in parameter space that a square object is present in the image? Give the main features of an algorithm to decide that a square object is present and to locate it.

d. Examine in detail whether an algorithm using the strategy described in (c) would become confused if (1) parts of some sides of the square were occluded; (2) one or more sides of the square were missing; (3) several squares appeared in the image; (4) several of these complications occurred together.

e. How important is it to this type of algorithm to have edge detectors that are capable of accurately determining edge orientation? Describe a type of edge detector that is capable of achieving this.

4. a. Describe the use of the HT for circular object detection, assuming that object size is known in advance. Show also how a method for detecting ellipses could be adapted for detecting circles of unknown size.

b. A new method is suggested for circle location which involves scanning the image both horizontally and vertically. In each case, the midpoints of chords are determined and their x or y coordinates are accumulated in separate 1-D histograms. Show that these can be regarded as simple types of HT, from which the positions of circles can be deduced. Discuss whether any problems would arise with this approach; consider also whether it would lead to any advantages relative to the standard HT for circle detection.

c. A further method is suggested for circle location. This again involves scanning the image horizontally, but in this case, for every chord that is found, an estimate is immediately made of the *two* points at which the center could lie, and votes placed at those locations. Work out the geometry of this method, and determine whether this method is faster than the method outlined in (b). Determine whether the method has any disadvantages compared with method in (b)?

5. Determine which of the methods described in this chapter will detect (1) hyperbolas, (2) curves of the form $Ax^3 + By^3 = 1$, (3) curves of the form $Ax^4 + Bx + Cy^4 = 1$.
6. Prove Eq. (10.18) for an ellipse. *Hint*: Write the coordinates of P and Q in suitable parametric forms, and then use the fact that OP$\perp$OQ to eliminate one of the parameters from the left-hand side of the equation.
7. Describe the *diameter bisection* and *chord−tangent* methods for the location of ellipses in images, and compare their properties. Justify the use of the chord−tangent method by proving its validity for circle detection and then extending the proof to cover ellipse detection.
8. Round coins of a variety of sizes are to be located, identified, and sorted in a vending machine. Discuss whether the chord−tangent method should be used for this purpose instead of the usual form of HT circle location scheme operating within a 3D (x, y, r) parameter space.
9. Outline each of the following methods for locating ellipses using the HT: (1) the *diameter bisection* method; (2) the *chord−tangent* method. Explain the principles on which these methods rely. Determine which is the more robust and compare the amounts of computation each requires.
10. For the diameter bisection method, searching through lists of edge points with the right orientations can take excessive computation. It is suggested that a two-stage approach might speed up the process: (1) load the edge points into a table which may be addressed by orientation; (2) look up the right edge points by feeding appropriate orientations into the table. Estimate how much this would be likely to speed up the diameter bisection method.
11. It is found that the diameter bisection method sometimes becomes confused when several ellipses appear in the same image, and generates false "centers" that are not situated at the centers of any ellipses. It is also found that certain other shapes are detected by the diameter bisection method. Ascertain in each case quite what the method is sensitive to, and consider ways in which these problems might be overcome.