Several additional features of the two *live* methods are worth mentioning. As was stressed earlier, design of border-detection cost functions often requires substantial experience and experimentation. To facilitate the method's use by non-experts, an automated approach has been developed that determines optimal border features from examples of the correct borders. Another automated step is available to specify optimal parameters of cost transforms to create a powerful cost function (Section 6.2.4). Consequently, the resultant optimal cost function is specifically designed for a particular application and can be conveniently obtained by presenting a small number of example border segments during the method's training stage. Additionally, the method can easily be applied to three-dimensional image data by incorporation of a cost element comparing the border positions in adjacent image slices.

6.2.6 Hough transforms

If an image consists of objects with known shape and size, segmentation can be viewed as a problem of finding this object within an image. Typical tasks are to locate circular pads in printed circuit boards, or to find objects of specific shapes in aerial or satellite data, etc. One of many possible ways to solve these problems is to move a mask with an appropriate shape and size along the image and look for correlation between the image and the mask, as discussed in Section 6.4. Unfortunately, the specified mask often differs too much from the object's representation in the processed data, because of shape distortions, rotation, zoom, etc. One very effective method that can solve this problem is the **Hough transform**, which can even be used successfully in segmentation of overlapping or semi-occluded objects.

To introduce the main concepts of the Hough transform, consider an example of circle detection. Let the task be to detect a dark circle of a known radius r in an image with a uniform bright background (shown in Figure 6.32a). The method starts with a search for dark image pixels; after such a pixel is found, a locus of potential center points of the circle associated with it can be determined. Such a locus of potential center points forms a circle with the radius r as demonstrated in Figure 6.32b. If the loci of potential circle centers are constructed for all dark pixels identified in the original image, the frequency can be determined with which each pixel of the image space occurs as an element of the circle-center loci. As seen from Figure 6.32c, the true center of the circle being sought is represented by the pixel with the highest frequency of occurrence in the circle-center loci. Thus, the center of the searched circle is determined. With the known circle radius, the image segmentation is complete. Figure 6.32d presents intuitive proof that the Hough transform can be successfully applied to images with incomplete information about the searched objects (a circle in our case) and/or in the presence of additional structures and noise. The remainder of this section describes the Hough transform methodology in detail.

The original Hough transform was designed to detect straight lines and curves [Hough, 1962], and this original method can be used if analytic equations of object borderlines are known—no prior knowledge of region position is necessary. A big advantage of this approach is robustness of segmentation results; that is, segmentation is not too sensitive to imperfect data or noise. Nevertheless, it is often impossible to get analytic expressions describing borders. Later, a generalized Hough transform will be described that can find objects even if an analytic expression of the border is not known.

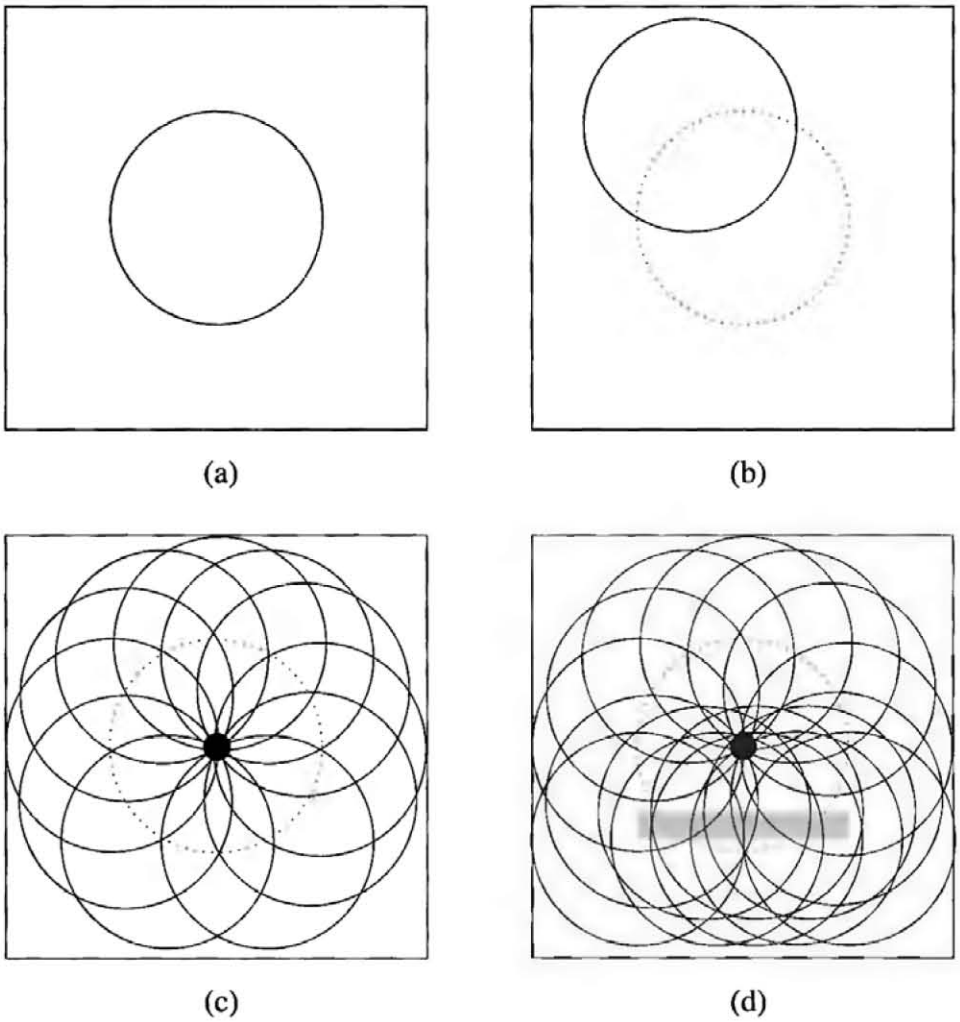


Figure 6.32: Hough transform—example of circle detection. (a) Original image of a dark circle (known radius r) on a bright background. (b) For each dark pixel, a potential circle-center locus is defined by a circle with radius r and center at that pixel. (c) The frequency with which image pixels occur in the circle-center loci is determined—the highest-frequency pixel represents the center of the circle (marked by •). (d) The Hough transform correctly detects the circle (marked by •) in the presence of incomplete circle information and overlapping structures. (See Figure 6.37 for a real-life example.)

The basic idea of the method can be seen from the simple problem of detecting a straight line in an image [Duda and Hart, 1972]. A straight line is defined by two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$ (shown in Figure 6.33a). All straight lines going through the point A are given by the expression $y_1 = kx_1 + q$ for some values of k and q. This means that the same equation can be interpreted as an equation in the parameter space k, q; all the straight lines going through the point A are then represented by the equation $q = -x_1 k + y_1$ (see Figure 6.33b). Straight lines going through the point B can likewise be represented as $q = -x_2 k + y_2$. The only common point of both straight lines in the k, q parameter space is the point which in the original image space represents the only existing straight line connecting points A and B.

This means that any straight line in the image is represented by a single point in the k, q parameter space and any part of this straight line is transformed into the same point. The main idea of line detection is to determine all the possible line pixels in the image, to transform all lines that can go through these pixels into corresponding points in the

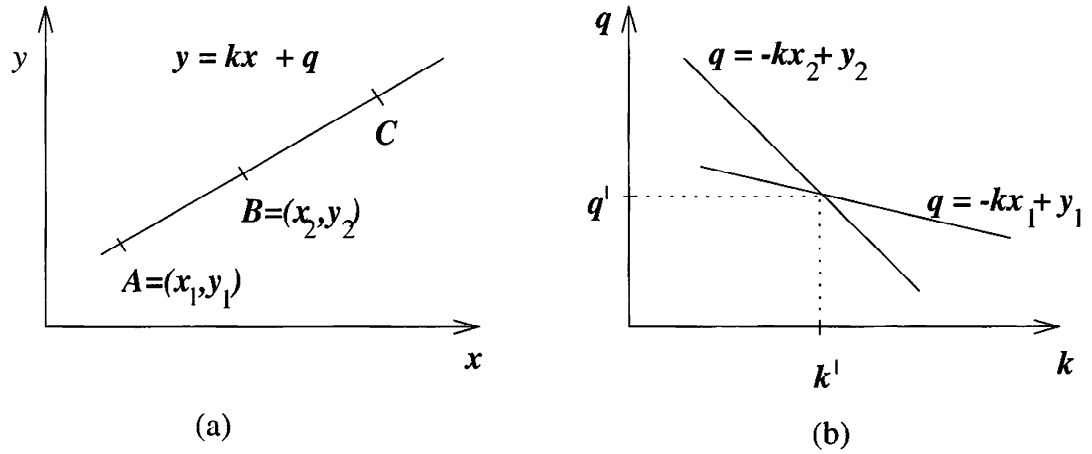


Figure 6.33: Hough transform principles. (a) Image space. (b) k, q parameter space.

parameter space, and to detect the points (a, b) in the parameter space that frequently resulted from the Hough transform of lines $y = ax + b$ in the image.

These main steps will be described in more detail. Detection of all possible line pixels in the image may be achieved by applying an edge detector to the image; then, all pixels with edge magnitude exceeding some threshold can be considered possible line pixels (referred to as edge pixels below). In the most general case, nothing is known about lines in the image, and therefore lines of any direction may go through any of the edge pixels. In reality, the number of these lines is infinite; however, for practical purposes, only a limited number of line directions may be considered. The possible directions of lines define a discretization of the parameter k. Similarly, the parameter q is sampled into a limited number of values. The parameter space is not continuous any more, but rather is represented by a rectangular structure of cells. This array of cells is called the **accumulator array** A, whose elements are **accumulator cells** $A(k, q)$. For each edge pixel, parameters k, q are determined which represent lines of allowed directions going through this pixel. For each such line, the values of line parameters k, q are used to increase the value of the accumulator cell $A(k, q)$. Clearly, if a line represented by an equation $y = ax + b$ is present in the image, the value of the accumulator cell $A(a, b)$ will be increased many times—as many times as the line $y = ax + b$ is detected as a line possibly going through any of the edge pixels. For any pixel P, lines going through it may have any direction k (from the set of allowed directions), but the second parameter q is constrained by the image co-ordinates of the pixel P and the direction k. Therefore, lines existing in the image will cause large values of the appropriate accumulator cells in the image, while other lines possibly going through edge pixels, which do not correspond to lines existing in the image, have different k, q parameters for each edge pixel, and therefore the corresponding accumulator cells are increased only rarely. In other words, lines existing in the image may be detected as high-valued accumulator cells in the accumulator array, and the parameters of the detected line are specified by the accumulator array co-ordinates. As a result, line detection in the image is transformed to detection of local maxima in the accumulator space.

It has been noted that an important property of the Hough transform is its insensitivity to missing parts of lines, to image noise, and to other non-line structures co-existing in the image. Insensitivity to data imprecision and noise can be seen in Figure 6.34. This is caused by the robustness of transformation from the image space into the accumulator

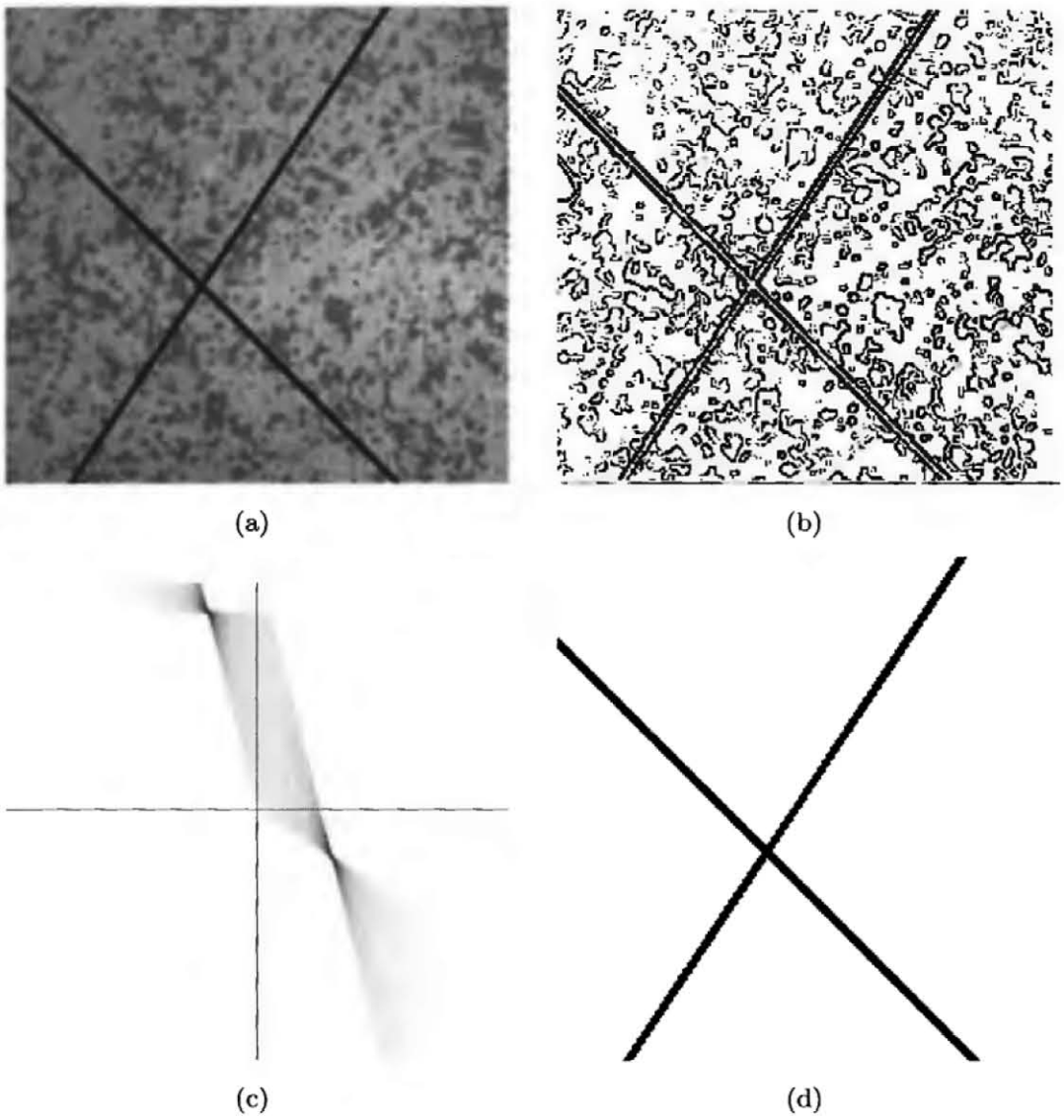


Figure 6.34: Hough transform—line detection. (a) Original image. (b) Edge image (note many edges, which do not belong to the line). (c) Parameter space. (d) Detected lines.

space—a missing part of the line will cause only a lower local maximum because a smaller number of edge pixels contributes to the corresponding accumulator cell. A noisy or only approximately straight line will not be transformed into a point in the parameter space, but rather will result in a cluster of points, and the cluster center of gravity can be considered the straight line representation.

Note that the parametric equation of the line $y = kx + q$ is appropriate only for explanation of the Hough transform principles—it causes difficulties in vertical line detection ($k \to \infty$) and in non-linear discretization of the parameter k. If a line is represented as

$$s = x \cos\theta + y \sin\theta \,, \tag{6.25}$$

the Hough transform does not suffer from these limitations. Again, the straight line is transformed to a single point (see Figure 6.35). A practical example showing the segmentation of an MR image of the brain into the left and right hemispheres is given in Figure 6.36.

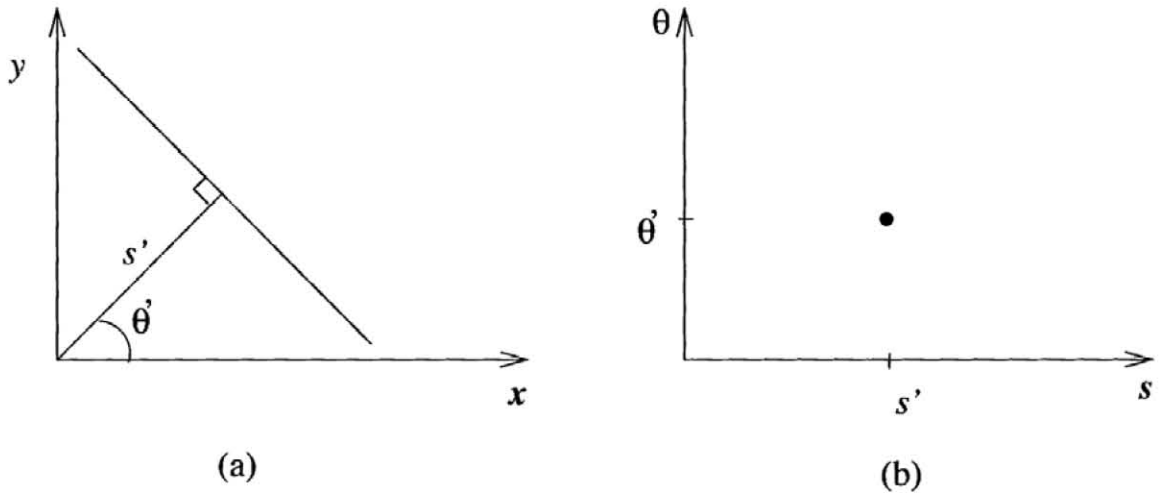
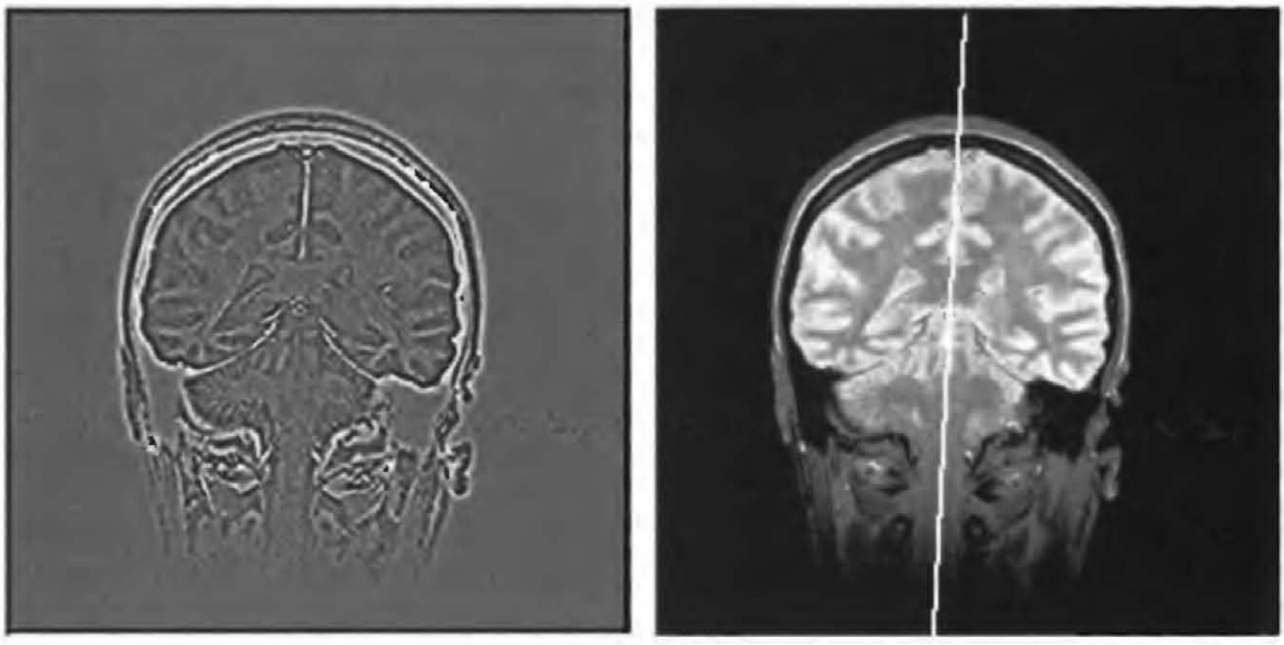


Figure 6.35: Hough transform in s, θ space. (a) Straight line in image space. (b) s, θ parameter space.

Discretization of the parameter space is an important part of this approach [Yuen and Hlavac, 1991]; also, detecting the local maxima in the accumulator array is a non-trivial problem. In reality, the resulting discrete parameter space usually has more than one local maximum per line existing in the image, and smoothing the discrete parameter space may be a solution. All these remarks remain valid if more complex curves are sought in the image using the Hough transform, the only difference being the dimensionality of the accumulator array.

Generalization to more complex curves that can be described by an analytic equation is straightforward. Consider an arbitrary curve represented by an equation $f(\mathbf{x}, \mathbf{a}) = 0$, where $\mathbf{a}$ is the vector of curve parameters.



(a) (b)

Figure 6.36: Hough transform line detection used for MRI brain segmentation to the left and right hemispheres. (a) Edge image. (b) Segmentation line in original image data.

Algorithm 6.14: Curve detection using the Hough transform

1. Quantize parameter space within the limits of parameters $\mathbf{a}$. The dimensionality n of the parameter space is given by the number of parameters of the vector $\mathbf{a}$.
2. Form an n-dimensional accumulator array $A(\mathbf{a})$ with structure matching the quantization of parameter space; set all elements to zero.
3. For each image point (x_1, x_2) in the appropriately thresholded gradient image, increase all accumulator cells $A(\mathbf{a})$ if $f(\mathbf{x}, \mathbf{a}) = 0$

$$A(\mathbf{a}) = A(\mathbf{a}) + \Delta A$$

   for all $\mathbf{a}$ inside the limits used in step 1.
4. Local maxima in the accumulator array $A(\mathbf{a})$ correspond to realizations of curves $f(\mathbf{x}, \mathbf{a})$ that are present in the original image.

If we are looking for circles, the analytic expression $f(\mathbf{x}, \mathbf{a})$ of the desired curve is

$$(x_1 - a)^2 + (x_2 - b)^2 = r^2 , \tag{6.26}$$

where the circle has center (a, b) and radius r. Therefore, the accumulator data structure must be three-dimensional. For each pixel $\mathbf{x}$ whose edge magnitude exceeds a given threshold, all accumulator cells corresponding to potential circle centers (a, b) are incremented in step 3 of the given algorithm. The accumulator cell $A(a, b, r)$ is incremented if the point (a, b) is at distance r from point $\mathbf{x}$, and this condition is valid for all triplets (a, b, r) satisfying equation (6.26). If some potential center (a, b) of a circle of radius r is frequently found in the parameter space, it is highly probable that a circle with radius r and center (a, b) really exists in the processed data.

The processing results in a set of parameters of desired curves $f(\mathbf{x}, \mathbf{a}) = 0$ that correspond to local maxima of accumulator cells in the parameter space; these maxima best match the desired curves and processed data. Parameters may represent unbounded analytic curves (e.g., line, ellipse, parabola, etc.), but to look for finite parts of these curves, the end points must be explicitly defined and other conditions must be incorporated into the algorithm. Even though the Hough transform is a very powerful technique for curve detection, exponential growth of the accumulator data structure with the increase of the number of curve parameters restricts its practical usability to curves with few parameters.

If prior information about edge directions is used, computational demands can be decreased significantly. Consider the case of searching the circular boundary of a dark region, letting the circle have a constant radius $r = R$ for simplicity. Without using edge direction information, all accumulator cells $A(a, b)$ are incremented in the parameter space if the corresponding point (a, b) is on a circle with center $\mathbf{x}$. With knowledge of direction, only a small number of the accumulator cells need be incremented. For example, if edge directions are quantized into eight possible values, only one-eighth of the circle need take part in incrementing of accumulator cells. Of course, estimates of edge direction are unlikely to be precise—if we anticipate edge direction errors of $\pi/4$, three-eighths of the circle will require accumulator cell incrementing. Using edge directions, candidates

for parameters a and b can be identified from the following formulae:

$$\begin{aligned} a &= x_1 - R\cos\big(\psi(\mathbf{x})\big)\,, \\ b &= x_2 - R\sin\big(\psi(\mathbf{x})\big)\,, \qquad \psi(\mathbf{x}) \in \big[\phi(\mathbf{x}) - \Delta\phi, \phi(\mathbf{x}) + \Delta\phi\big]\,, \end{aligned} \tag{6.27}$$

where $\phi(\mathbf{x})$ refers to the edge direction in pixel $\mathbf{x}$ and $\Delta\phi$ is the maximum anticipated edge direction error. Accumulator cells in the parameter space are then incremented only if (a, b) satisfy equation (6.27). Another heuristic that has a beneficial influence on the curve search is to weight the contributions to accumulator cells $A(\mathbf{a})$ by the edge magnitude in pixel $\mathbf{x}$; thus the increment ΔA in step 3 of Algorithm 6.14 $[A(\mathbf{a}) = A(\mathbf{a}) + \Delta A]$ will be greater if it results from the processing of a pixel with larger edge magnitude. Figure 6.37 demonstrates circle detection when circular objects of known radius overlap and the image contains many additional structures causing the edge image to be very noisy. Note that the parameter space with three local maxima corresponding to centers of three circular objects.

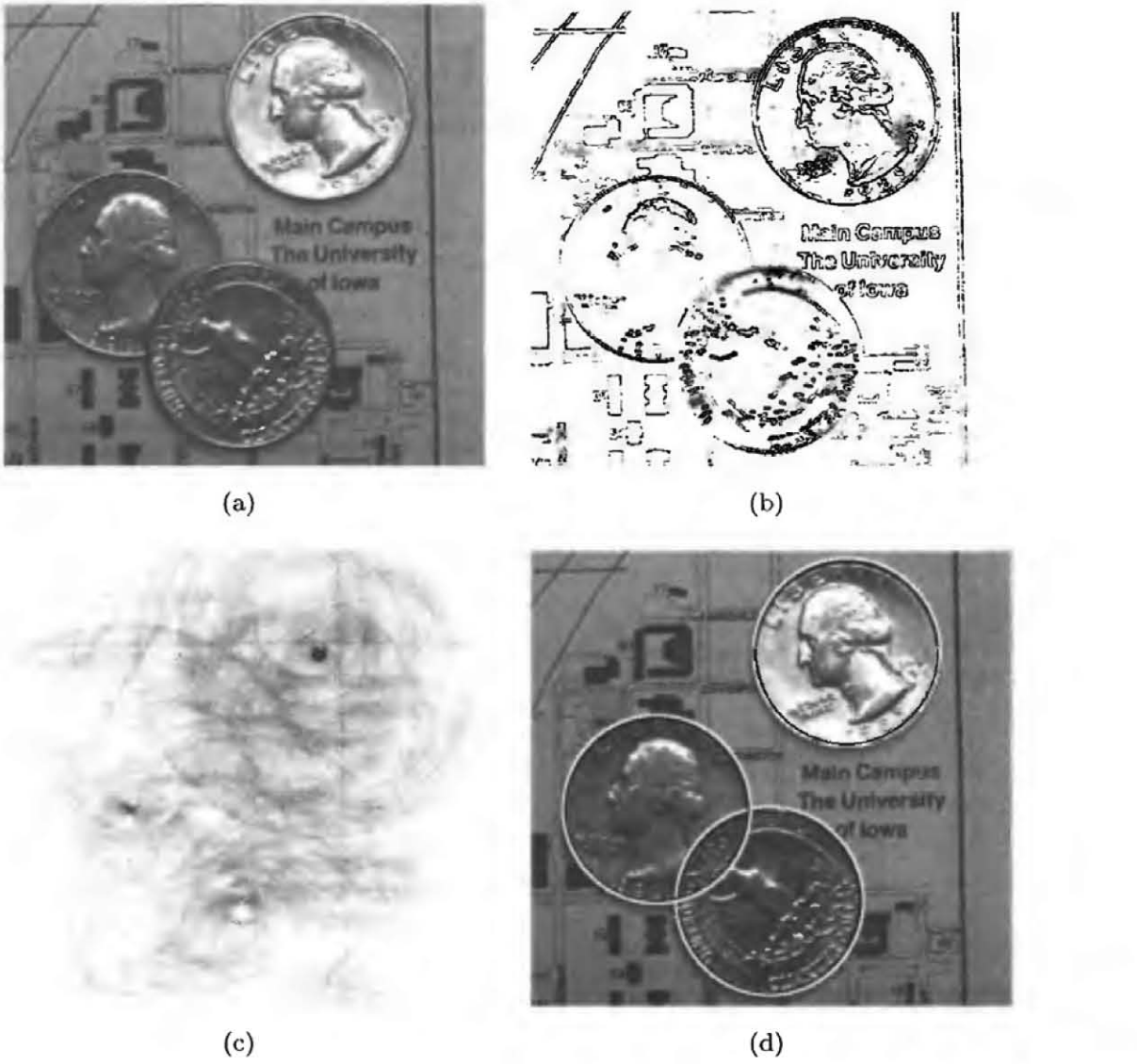


Figure 6.37: Hough transform—circle detection. (a) Original image. (b) Edge image (note that the edge information is far from perfect). (c) Parameter space. (d) Detected circles.

The randomized Hough transform offers a different approach to achieve increased efficiency [Xu and Oja, 1993]; it randomly selects n pixels from the edge image and determines n parameters of the detected curve followed by incrementing a single accumulator cell only. Extensions to the randomized Hough transform use local information about the edge image and apply the Hough transform process to a neighborhood of the edge pixel [Kalviainen et al., 1995].

If the parametric representations of the desired curves or region borders are known, this method works very well, but unfortunately this is not often the case. The desired region borders can rarely be described using a parametric boundary curve with a small number of parameters; in this case, a generalized Hough transform [Ballard, 1981; Davis, 1982; Illingworth and Kittler, 1987] can offer the solution. This method constructs a parametric curve (region border) description based on sample situations detected in the learning stage. Assume that shape, size, and rotation of the desired region are known. A reference point $\mathbf{x}^R$ is chosen at any location inside the sample region, then an arbitrary line can be constructed starting at this reference point aiming in the direction of the region border (see Figure 6.38). The border direction (edge direction) is found at the intersection of the line and the region border. A reference table (referred to as the R-table in [Ballard, 1981]) is constructed, and intersection parameters are stored as a function of the border direction at the intersection point; using different lines aimed from the reference point, all the distances of the reference point to region borders and the border directions at the intersections can be found. The resulting table can be ordered according to the border directions at the intersection points. As Figure 6.38 makes clear, different points $\mathbf{x}$ of the region border can have the same border direction, $\phi(\mathbf{x}) = \phi(\mathbf{x}')$. This implies that there may be more than one (r, α) pair for each ϕ that can determine the co-ordinates of a potential reference point.

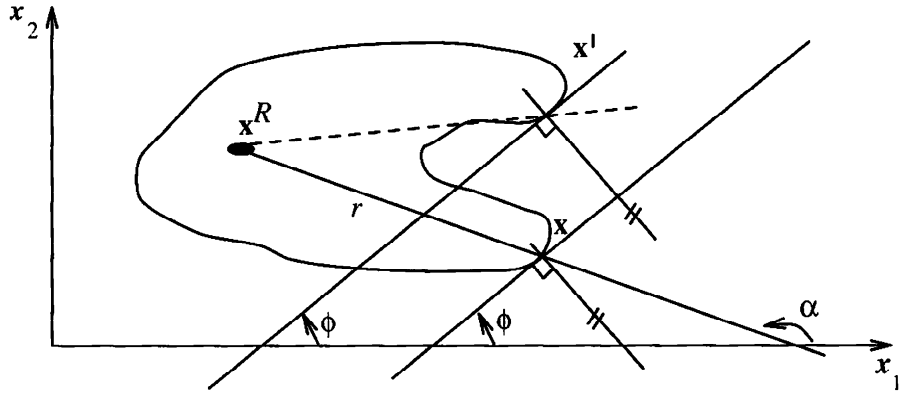


Figure 6.38: Principles of the generalized Hough transform: geometry of R-table construction.

An example of an R-table is given in Table 6.1. Assuming no rotation and known size, remaining description parameters required are the co-ordinates of the reference point (x_1^R, x_2^R). If size and rotation of the region may vary, the number of parameters increases to four. Each pixel $\mathbf{x}$ with a significant edge in the direction $\phi(\mathbf{x})$ has co-ordinates of potential reference points $\{x_1 + r(\phi)\cos\big(\alpha(\phi)\big), x_2 + r(\phi)\sin\big(\alpha(\phi)\big)\}$. These must be computed for all possible values of r and α according to the border direction $\phi(\mathbf{x})$ given in the R-table. The following algorithm presents the generalized Hough transform in the most general of cases in which rotation (τ) and size (S) may both change. If either there is no change in rotation ($\tau = 0$), or there is no size change ($S = 1$), the resulting accumulator data structure A is simpler.

ϕ_1	$(r_1^1, \alpha_1^1), (r_1^2, \alpha_1^2), \ldots, (r_1^{n_1}, \alpha_1^{n_1})$
ϕ_2	$(r_2^1, \alpha_2^1), (r_2^2, \alpha_2^2), \ldots, (r_2^{n_2}, \alpha_2^{n_2})$
ϕ_3	$(r_3^1, \alpha_3^1), (r_3^2, \alpha_3^2), \ldots, (r_3^{n_3}, \alpha_3^{n_3})$
$\ldots$	$\ldots$
ϕ_k	$(r_k^1, \alpha_k^1), (r_k^2, \alpha_k^2), \ldots, (r_k^{n_k}, \alpha_k^{n_k})$

Table 6.1: R-table

Algorithm 6.15: Generalized Hough transform

1. Construct an R-table description of the desired object.
2. Form a data structure A that represents the potential reference points

$$A(x_1, x_2, S, \tau) .$$

   Set all accumulator cell values $A(x_1, x_2, S, \tau)$ to zero.
3. For each pixel (x_1, x_2) in a thresholded gradient image, determine the edge direction $\Phi(\mathbf{x})$; find all potential reference points $\mathbf{x}^R$ and increase all

$$A(\mathbf{x}^R, S, \tau) = A(\mathbf{x}^R, S, \tau) + \Delta A$$

   for all possible values of rotation and size change

$$x_1^R = x_1 + r(\phi + \tau)\, S \cos\big(\alpha(\phi + \tau)\big) ,$$
$$x_2^R = x_2 + r(\phi + \tau)\, S \sin\big(\alpha(\phi + \tau)\big) .$$

4. The location of suitable regions is given by local maxima in the A data structure.

The Hough transform was initially developed to detect analytically defined shapes, such as lines, circles, or ellipses in general images, and the generalized Hough transform can be used to detect arbitrary shapes. However, even the generalized Hough transform requires the complete specification of the exact shape of the target object to achieve precise segmentation. Therefore, it allows detection of objects with complex but pre-determined, shapes. Other varieties exist that allow detection of objects whose exact shape is unknown, assuming a priori knowledge can be used to form an approximate model of the object.

The Hough transform has many desirable features [Illingworth and Kittler, 1988]. It recognizes partial or slightly deformed shapes, therefore behaving extremely well in recognition of occluded objects. It may be also used to measure similarity between a model and a detected object on the basis of size and spatial location of peaks in the parameter space. The Hough transform is very robust in the presence of additional structures in the image (other lines, curves, or objects) as well as being insensitive to image noise. Moreover, it may search for several occurrences of a shape during the same processing pass. Unfortunately, the conventional sequential approach requires a lot of storage and extensive computation. However, its inherent parallel character gives the potential for real-time implementations.

Many serious implementational problems were only touched upon here (shape parameterization, accumulation in parameter space, peak detection in parameter space, etc.). Details are discussed in surveys [Illingworth and Kittler, 1988; Princen et al., 1994], where extensive lists of references may also be found. Because of the large time requirements in the sequential version, effort has been devoted to hierarchical approaches [Neveu, 1986; Princen et al., 1989]; fast algorithms were developed [Guil et al., 1995]; gray-scale Hough transforms working directly in image data were presented in [Lo and Tsai, 1995]; methods combining the Hough transform and automated line tracing were studied [Wang and Howarth, 1989; Lerner and Morelli, 1990], and many parallel implementations were tested [Oyster, 1987; Olariu et al., 1993; Chung and Lin, 1995]. The unique properties of the Hough transform also provoke more and more applications [Illingworth and Kittler, 1988; McKenzie and Protheroe, 1990; Brummer, 1991]. Features of many existing varieties of the Hough transform, together with performance comparisons, are given in [Kalviainen et al., 1995].

6.2.7 Border detection using border location information

If any information about boundary location or shape is known, it is of benefit to use it. The information may, for instance, be based on some higher-level knowledge, or can result from segmentation applied to a lower-resolution image.

One possibility is to determine a boundary in an image as the location of significant edges positioned close to an assumed border if the edge directions of these significant edges match the assumed boundary direction. The new border pixels are sought in directions perpendicular to the assumed border (see Figure 6.39). If a large number of border elements satisfying the given conditions is found, an approximate curve is computed based on these pixels, and a new, more accurate, border results.

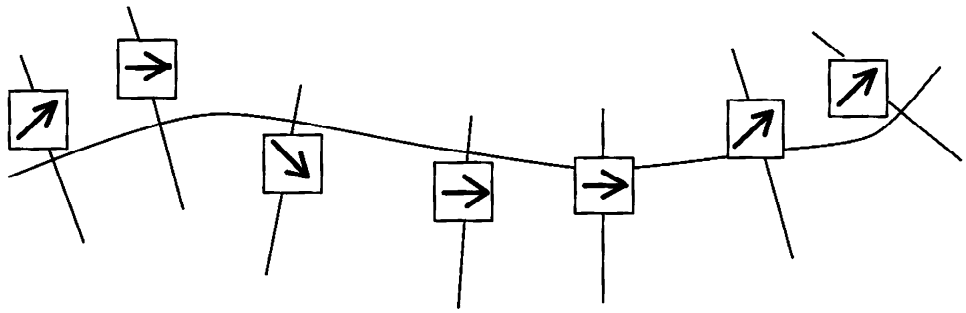


Figure 6.39: A priori information about boundary location.

Another possibility is based on prior knowledge of end points—this approach assumes low image noise and relatively straight boundaries. The process iteratively partitions the border and searches for the strongest edge located on perpendiculars to the line connecting end points of each partition; perpendiculars are located at the center of the connecting straight line—see Figure 6.40. The strongest significant edge located on the perpendicular that is close to the straight line connecting the end points of the current partition is accepted as a new border element. The iteration process is then repeated.

Another approach to contour detection has been introduced in [Kass et al., 1987] in which active contour models (snakes) start their search for a contour taking advantage of user-provided knowledge about approximate position and shape of the required contour.

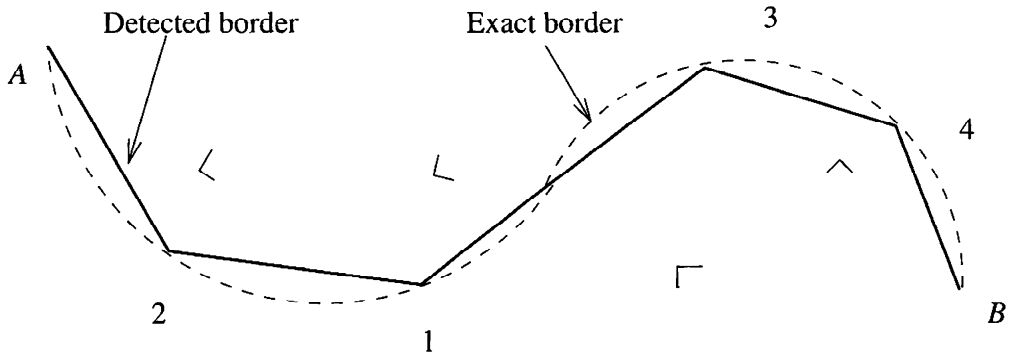


Figure 6.40: Divide-and-conquer iterative border detection; numbers show the sequence of division steps.

An optimization method refines the starting contour estimate and matches the desired contour. This approach is discussed in Section 7.2.

6.2.8 Region construction from borders

All methods considered hitherto have focused on the detection of borders that partially or completely segmented the processed image. If a complete segmentation is achieved, the borders segment an image into regions; but if only a partial segmentation results, regions are not defined uniquely and region determination from borders may be a very complex task requiring cooperation with higher-level knowledge. However, methods exist that are able to construct regions from partial borders which do not form closed boundaries. These methods do not always find acceptable regions, but they are useful in many practical situations.

One of them is the **superslice** method [Milgram, 1979], which is applicable if regions have dominant gray-level properties. The approach assumes that some border-part locations are known in the image; the image data is then thresholded using different thresholds. Regions resulting from the thresholding for which the detected boundaries best coincide with assumed boundary segments are then accepted as correct.

Better results can be obtained by applying a method described in [Hong et al., 1980] based on the existence of partial borders in the processed image. Region construction is based on probabilities that pixels are located inside a region closed by the partial borders. The border pixels are described by their positions and by pixel edge directions $\phi(\mathbf{x})$. The closest 'opposite' edge pixel is sought along a perpendicular to each significant image edge, and then closed borders are constructed from pairs of opposite edge pixels. A pixel is a potential region member if it is on a straight line connecting two opposite edge pixels. The final decision on which of the potential region pixels will form a region is probabilistic.

Algorithm 6.16: Region forming from partial borders

1. For each border pixel $\mathbf{x}$, search for an opposite edge pixel within a distance not exceeding a given maximum M. If an opposite edge pixel is not found, process the next border pixel in the image. If an opposite edge pixel is found, mark each pixel on the connecting straight line as a potential region member.

2. Compute the number of markers for each pixel in the image (the number of markers tells how often a pixel was on a connecting line between opposite edge pixels). Let $b(\mathbf{x})$ be the number of markers for the pixel $\mathbf{x}$.
3. The weighted number of markers $B(\mathbf{x})$ is then determined as follows:

$$\begin{aligned} B(\mathbf{x}) &= 0.0 \quad \text{for } b(\mathbf{x}) = 0\,, \\ &= 0.1 \quad \text{for } b(\mathbf{x}) = 1\,, \\ &= 0.2 \quad \text{for } b(\mathbf{x}) = 2\,, \\ &= 0.5 \quad \text{for } b(\mathbf{x}) = 3\,, \\ &= 1.0 \quad \text{for } b(\mathbf{x}) > 3\,. \end{aligned} \tag{6.28}$$

The confidence that a pixel $\mathbf{x}$ is a member of a region is given as the sum $\sum_i B(\mathbf{x_i})$ in a 3×3 neighborhood of the pixel $\mathbf{x}$. If the confidence that a pixel $\mathbf{x}$ is a region member is one or larger, then pixel $\mathbf{x}$ is marked as a region pixel, otherwise it is marked as a background pixel.

Note that this method allows the construction of bright regions on a dark background as well as dark regions on a bright background by taking either of the two options in the search for opposite edge pixels—step 1. Search orientation depends on whether relatively dark or bright regions are constructed. If $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ are directions of edges, the condition that must be satisfied for $\mathbf{x}$ and $\mathbf{y}$ to be opposite is

$$\frac{\pi}{2} < \left|(\phi(\mathbf{x}) - \phi(\mathbf{y})) \bmod (2\pi)\right| < \frac{3\pi}{2}\,. \tag{6.29}$$

Note that it is possible to take advantage of prior knowledge of maximum region sizes—this information defines the value of M in step 1 of the algorithm, the maximum search length for the opposite edge pixel.

This method was applied to form texture primitives (Chapter 15 [Hong et al., 1980]) as shown in Figure 6.41. The differences between the results of this region detection method and those obtained by thresholding applied to the same data are clearly visible if Figures 6.41b and 6.41c are compared.

6.3 Region-based segmentation

The aim of the segmentation methods described in the previous section was to find borders between regions; the following methods construct regions directly. It is easy to construct regions from their borders, and it is easy to detect borders of existing regions. However, segmentations resulting from edge-based methods and region-growing methods are not usually exactly the same, and a combination of results may often be a good idea. Region growing techniques are generally better in noisy images, where borders are extremely difficult to detect. Homogeneity is an important property of regions and is used as the main segmentation criterion in region growing, whose basic idea is to divide an image into zones of maximum homogeneity. The criteria for homogeneity can be based on gray-level, color, texture, shape, model (using semantic information), etc. Properties