Pattern ReuKfninon VoJ 13, No. 2. pp. Ill 122. 1981
Printed in Grcal Britain.

0031-3203/81/020111-12 $02.00/0
Pergamon Press Ltd.
£> Pattern Recognition Society

GENERALIZING THE HOUGH TRANSFORM TO DETECT ARBITRARY SHAPES*

D. H. BALLARD Computer Science Department,
University of Rochester, Rochester, NY 14627, U.S.A.

***(Received* 10 *October* 1979; *in revised form* 9 *September* 1980; *received for publication* 23 *September* 1980)**

Abstract—The Hough transform is a method for Meeting curves by exploiting the duality between points on a curve and parameters of that curve. The initial work showed how to detect both analytic curves"-*' and non-analytic curves,[151] but these methods were restricted to binary edge images. This work was generalized to the detection of some analytic curves in grey level images, specifically lines,'*' circles[131] and parabolas.'[61] The line detection case is the best known of these and has been ingeniously exploited in several applications.'[7]"" We show how the boundaries of an *arbitrary* non-analytic shape can be used to construct a mapping between image space and Hough transform space. Such a mapping can be exploited to detect instances of that particular shape in an image. Furthermore, variations in the shape such as rotations, scale changes or figure ground reversals correspond to straightforward transformations of this mapping. However, the most remarkable property is that such mappings can be composed to build mappings for complex shapes from the mappings of simpler component shapes. This makes the generalized Hough transform a kind of universal transform which can be used to find arbitrarily complex shapes.

Image processing Hough transform Shape recognition Pattern recognition Parallel algorithms

I. INTRODUCTION

In an image, the pertinent information about an object is very often contained in the shape of its boundary. Some appreciation of the importance of these boundary shapes in human vision can be gained from experiments performed on the human visual system, which have shown that crude encodings of the boundaries are often sufficient for object recognition'[10]' and that the image may be initially encoded as an 'edge image', i.e. an image of local intensity or color gradients. Marr'[1]" has termed this edge image a 'primal sketch' and suggested that this may be a necessary first step in image processing. We describe a very general algorithm for detecting objects of a specified shape from an image that has been transformed into such an edge representation. In that representation, sample points in the image no longer contain grey level information, but instead each sample point contains a magnitude and direction representing the severity and orientation of the local grey level change.

Operators that transform the image in such a way are known as edge operators, and many such operators are available, all based on different models of the local grey level changes. Two of the most used are the gradient operator (for example, see Prewitt'[12]') and the Hueckel operator,"[3]' which model local grey level changes as a ramp and a step respectively.

*** The research described in this report was supported in part by NIH Grant R23-HL-2153-01 and in part by the Alfred P. Sloan Foundation Grant 78-4-15.**

Our generalized Hough algorithm uses edge information to define a mapping from the orientation of an edge point to a reference point of the shape. The reference point may be thought of as the origin of a local co-ordinate system for the shape. Then there is an easy way of computing a measure which rates how well points in the image are likely to be origins of the specified shape. Figure 1 shows a few graphic examples of the information used by the generalized Hough transform. Lines indicate gradient directions. A feature of the transform is that it will work even when the boundary is disconnected due to noise or occlusions. This is generally not true for other strategies which track edge segments.

The original algorithm by Hough'[2]' did not use

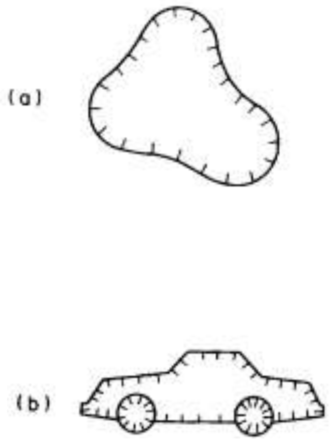


Fig. 1. Kinds of shapes detected with generalized Hough transform, (a) Simple shape; (b) composite shape.

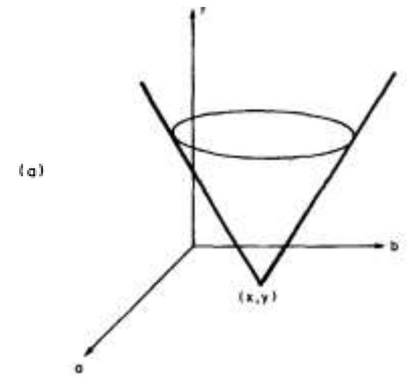
orientation information of the edge, and was considerably inferior to later work using the edge orientation for parametric curves.'5-6'141 Shapiro"516 17> has collected a good bibliography of previous work as well as having contributed to the error analysis of the technique.

1.1 *Organization*

Section 2 describes the Hough transform for analytic curves. As an example of the parametric version of the transform, we use the ellipse. This example is very important due to the pervasiveness of circles in images, and the fact that a circle becomes an ellipse when rotated about an axis perpendicular to the viewing angle. Despite the importance of ellipses, not much work has used the Hough transform. The elliptical transform is discussed in detail in Section 3. Section 4 describes the generalized algorithm and its properties. Section S describes special strategies for implementing the algorithm and Section 6 summarizes its advantages.

2. THE HOUGH TRANSFORM FOR ANALYTIC CURVES

We consider analytic curves of the form *f(x,*a) = 0 where x is an image point and a is a parameter vector.



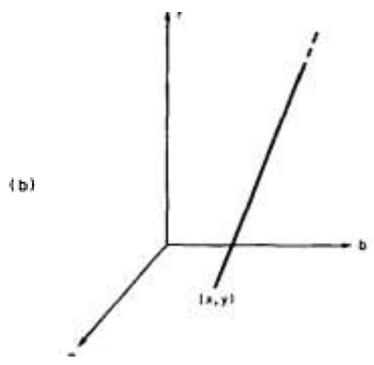


Fig. 2. (a) Locus of parameters with no directional information. (b) Locus of parameters with directional information.

To see how the Hough transform works for such curves, let us suppose we are interested in detecting circular boundaries in an image. In Cartesian coordinates, the equation for a circle is given by

$$(x-a)^2 + (y-b)^2 = r^2. \qquad (1)$$

Suppose also that the image has been transformed into an edge representation so that only the magnitude of local intensity changes is known. Pixels whose magnitude exceeds some threshold are termed *edge pixels.* For each edge pixel, we can ask the question: if this pixel is to lie on a circle, what is the locus for the parameters of that circle? The answer is a right circular cone, as shown in Fig. 2(a). This can be seen from equation (1) by treating x and *y* as fixed and letting, *a, b,* and r vary.

The interesting result about this locus in parameter space is the following. If a set of edge pixels in an image are arranged on a circle with parameters a_0, b_0, and r_0, the resultant loci of parameters for each such point will pass through the same point (a_0,b_0,r_0) in parameter space. Thus many such right circular cones will intersect at a common point.

2.1 *Directional information*

We see immediately that if we also use the *directional* information associated with the edge, this reduces the parameter locus to a line, as shown in Fig. 2(b). This is because the center of the circle for the point (x, y) must lie *r* units along the direction of the gradient. Formally, the circle involves 3 parameters. By using the equation for the circle together with its derivative, the number of free parameters is reduced to one. Formally, what happens is the equation

$$\frac{df}{dx}(\mathbf{x}, \mathbf{a}) = 0$$

introduces a term *Ay*/dx which is known since

$$\frac{dy}{dx} = \tan\left[\phi(\mathbf{x}) - \frac{\pi}{2}\right]$$

where *ф(x)* is the gradient direction. This suggests the following algorithm.

Hough algorithm for analytic curves in grey level images. For a specific curve/(x, a) = 0 with parameter vector a, form an array /1(a), initially set to zero. This array is termed an accumulator array. Then for each edge pixel x, compute all a such that *f{\, a)* = 0 and d//dx(x, a) = 0 and increment the corresponding accumulator array entries:

$$A(a) := A(я) + 1.$$

After each edge pixel x has been considered, local maxima in the array *A* correspond to curves of/in the image.

If only the equation/*(x, a) = 0 is used, the cost of the computation is exponential in the number of parameters minus one, that is, where *m* parameters each have *M* values, the computation is proportional to

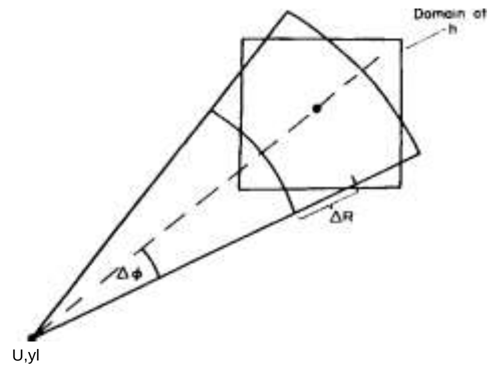


Fig. 3. Using convolution templates to compensate for errors.

M™ ~'. This is because the equation of the curve can be used to determine the last parameter. The use of gradient directional information saves the cost of another parameter making the total effort proportional to Af""^{2}, for $m > 2$.

2.2 *Compensating for errors*

A problem arises in detecting maxima in the array /1(a). Many sources of error effect the computation of the parameter vector a so that in general many array locations in the vicinity of the ideal point a are incremented instead of the point itself. One way of handling this problem is to use a formal error model on the incrementation step. This model would specify a set of nearby points instead of a single point. Shapiro'$^{1\ s\_18)}$ has done extensive work on this subject. Another solution to this problem is to replace uncom-pensated accumulator values by a function of the values themselves and nearby points after the incrementation step. The effect of this operation is to smooth the accumulator array. We show that, under the assumption of isotropic errors, these methods are equivalent.

Returning to the initial example of detecting circles, the smoothing of the accumulator array is almost equivalent to the change in the incrementing procedure we would use to allow for uncertainties in the gradient direction ϕ and the radius r. If we recognized these uncertainties as:

$$\phi(\mathbf{x}) \pm \Delta\phi$$
$$r \pm \Delta r(r)$$

we would increment all values of a which fall within the shaded band of Fig. 3. We let Ar increase with r so that uncertainties are counted on a percentage basis. Figure 3 shows the two-dimensional analog of the general three-dimensional case.

Suppose we approximate this procedure by incrementing all values of a which fall inside the square domain centered about the nominal center shown in Fig. 3, according to some point spread function *h*. After the first contributing pixel which increments center a_0 has been taken into account, the new accumulator array contents *A* will be given by JI(.) = *(■-■,,) (2)

where a = (a_1(a_2,r) and a_0 = {a_{l0},a_{20},r_0}. If we include all the contributing pixels for that center, denoted by C, the accumulator is

$$A(\mathbf{a}) = C(\mathbf{a}_0)h(\mathbf{a}-\mathbf{a}_0). \tag{3}$$

Finally for all incremented centers, we sum over a_0:
Thus within the approximation of letting the square represent the shaded band shown in Fig. 3, the

$$A(\mathbf{a}) = \sum_{\mathbf{a}_0} C(\mathbf{a}_0)h(\mathbf{a}-\mathbf{a}_0). \tag{4}$$

But $C(\mathbf{a}_0) = A(\mathbf{a}_0)$, so that

$$\begin{aligned} A(\mathbf{a}) &= \sum_{\mathbf{a}_0} A(\mathbf{a}_0)h(\mathbf{a}-\mathbf{a}_0) \\ &= A * h \\ &\equiv A_s(\mathbf{a}). \end{aligned} \tag{5}$$

smoothing procedure is equivalent to an accommodation for uncertainties in the gradient direction and radius.

3. AN EXAMPLE: ELLIPSES

The description of the algorithm in Section 2.1 is very terse and its implementation often requires considerable algebraic manipulation. We use the example of finding ellipses to show the kinds of calculation which must be done. Ellipses are an important example, as circles, which are a ubiquitous part of many everyday objects, appear as ellipses when viewed from a distant, oblique angle.
We use the center of the ellipse as a reference point and assume that it is centered at x_0, y_0 with major and minor diameters *a* and *b*. For the moment, we will assume that the ellipse is oriented with its major axis parallel to the x-axis. Later we will relax this requirement by introducing an additional parameter for arbitrary orientations. For the moment, assume *a* and *b* are fixed. Then the equation of the ellipse is:

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1. \tag{6}$$

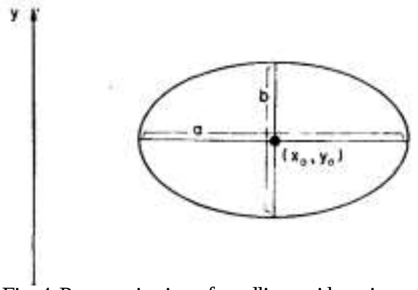


Fig. 4. Parametrization of an ellipse with major axis parallel to x-axis.

cal he

he ■-
ch to

Let $X = x - x_0$, $Y = y - y_0$, then

$$\frac{X^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{7}$$

Differentiating with respect to X

$$\frac{2X}{a^2} + \frac{2Y}{b^2}\,\frac{dY}{dX} = 0. \tag{8}$$

But dY/dX is known from the edge pixel information! Let $dY/dX = \xi$, then from (8)

$$X^2 = \left(\frac{a^2}{b^2}\xi\right)^2 Y^2. \tag{9}$$

Substituting in (7)

$$\frac{Y^2}{b^2}\left(1 + \frac{a^2}{b^2}\xi^2\right) = 1 \tag{10}$$

$$Y = \pm\frac{b^2}{\sqrt{\left(1 + \frac{a^2}{b^2}\xi^2\right)}} \tag{11}$$

so that

$$X = \pm\frac{a^2}{\sqrt{\left(1 + \frac{b^2}{a^2\xi^2}\right)}} \tag{12}$$

and finally, given *a.b.x.y* and d *Y/dX,* we can determine x_0 and y_0 as:

$$x_0 = x \pm \frac{a^2}{\sqrt{\left(1 + \frac{b^2}{a^2\xi^2}\right)}} \tag{13}$$

$$y_0 = y \pm \frac{b^2}{\sqrt{\left(1 + \frac{a^2\xi^2}{b^2}\right)}}. \tag{14}$$

The four solutions correspond to the four quadrants, as shown in Fig. 5. The appropriate quadrant can be found from the gradient by testing the signed differences *dY* and *dX.*

The final step is to handle rotations by introducing a fifth parameter 0. For an arbitrary 0, we calculate *(X, Y)* using

$$\xi = \tan\left(\phi - \theta - \frac{\pi}{2}\right)$$

and rotate these *(X, Y)* by *в* to obtain the correct

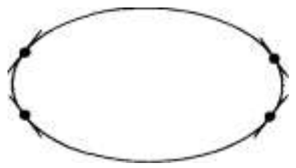


Fig. 5. Four reference point solutions resolvable with gradient quadrant information.

(x_0, y_0). In ALGOL we would implement this as:

```
procedure HoughEllipse (integer $X_{min}X_{max}$, $Y_{min}Y_{max}$,
$\theta_{min}\theta_{max}$, $a_{min}a_{max}$, $b_{min}b_{max}$, $x$, $y$, $x_0$, $y_0$, dx, dy; real angle, $\xi$;
integer array $A$, $P$);
begin;
for $x := x_{min}$ step dx to $x_{max}$ do
for $y := y_{min}$ step dy to $y_{max}$ do
  begin
    $dX := P(x + \text{delta}, y) - P(x, y)$;
    $dY := P(x, y + \text{delta}) - P(x, y)$;
      for $a := a_{min}$ step $da$ until $a_{max}$ do
      for $b := b_{min}$ step $db$ until $b_{max}$ do
      for $\theta := \theta_{min}$ step $d\theta$ until $\theta_{max}$ do
        begin;
          angle: $= \arctan\left(\frac{dY}{dX}\right) - \theta - \frac{\pi}{2}$;
          $\xi :=$ tan(angle);
          dx: $=$ Sign $X(dX, dY)\frac{a^2}{\sqrt{\left(1 + \frac{b^2}{a^2\xi^2}\right)}}$;
          dy: $=$ Sign Y $(dX, dY)\frac{b^2}{\sqrt{\left(1 + \frac{a^2}{b^2}\right)^2}}$;
          Rotate-by-Theta(dx, dy);
          $x_0 := x + dx$;
          $y_0 := y + dy$;
          $A(x_0, y_0, \theta, a, b) := A(x_0, y_0, \theta, a, b) + 1$;
        end;
end.
```

Notice that to determine the appropriate formulae for an arbitrary orientation angle *в,* we need only rotate the gradient angle and the offsets dx and dy. SignX and Sign У are functions which return ± 1 depending on the quadrant determined by *dX* and *dY.*

3.1 *Parameter space image space trade-offs*

Tsuji and Matsumoto"[9]' recognized that a decreased computational effort in parameter space could be traded for an increased effort in edge space. It is our intent to place these ideas on a formal footing. Later we will see that the same kind of trade-off is potentially available for the case of arbitrary shapes, but is impractical to implement.

An ellipse has five parameters. Referring to the basic algorithm in Section 2.1, we use the equation for the ellipse together with its derivative to solve for two of these parameters as a function of the other three. Thus the algorithm examines every edge point and uses a three-dimensional accumulator array so that the computations are of order $0(ed^3)$. Here *e* is the number of edge pixels and we are assuming *d* distinct values for each parameters. Suppose we use pairs of edge points in the algorithm. This results in four equations, two involving the equation for an ellipse evaluated at the different points and two for the related derivatives. This leaves one free parameter. Thus the resultant computational effort is now $0(e^2d)$. The detailed derivation of this form of the Hough algorithm is presented in the Appendix.

If parameter space can be highly constrained so that the set of plausible values is small, then the former technique will be more efficient, whereas if there are

Table 1. Analytic curves described in terms of the generalized shape parameters x_r, y_r, S_x, S_y, θ

Analytic form	Parameters	Equation
Line	S, θ	$x \cos\theta + y \sin\theta = S$
Circle	x_r, y_r, S	$(x - x_r)^2 + (y - y_r)^2 = S^2$
Parabola	x_r, y_r, S_x, θ	$(y - y_r)^2 = 4S_x(x - x_r)$*
Ellipse	$x_r, y_r, S_x, S_y, \theta$	$\frac{(y-y_r)^2}{S_y^2} + \frac{(x-x_r)^2}{S_x^2} = 1$*

* Plus rotation by θ.

relatively few edges and large variations in parameters, the latter will be more efficient.

4. GENERALIZING THE HOUGH TRANSFORM

To generalize the Hough algorithm to non-analytic curves we define the following parameters for a generalized shape:

$$\mathbf{a} = \{\mathbf{y}, \mathbf{s}, \theta\},$$

where y = (x$_r$, *y$_T$)* is a reference origin for the shape, *в* is its orientation, and s = *(s,, s$_y$)* describes two orthogonal scale factors. As before, we will provide an algorithm for computing the best set of parameters я for a given shape from edge pixel data. These parameters no longer have equal status. The reference origin location, y, is described in terms of a table of possible edge pixel orientations. The computation of the additional parameters s and *в* is then accomplished by straightforward transformations to this table. [To simplify the development slightly, and because of its practical significance, we will work with the four-dimensional sunspacea = *(y,s,в),* wheres is a scalar] In a sense this choice of parameters includes the previous analytic forms to which the Hough transform has been applied. Table 1 shows these relationships.

4.1 *Earlier work: arbitrary shapes in binary edge images*

Merlin and Farber[13] showed how to use a Hough algorithm when the desired curves could not be described analytically. Each shape must have a specific reference point. Then we can use the following algorithm for a shape with boundary points *B* denoted by {x$_B$} which are relative to some reference origin y.

Merlin-Farber Hough algorithm: non-analytic curves with no gradient direction information я = y. Form a two-dimensional accumulator array *A(л)* initialized to zero. For each edge pixel x and each boundary point *x$_B$*, compute a such that я = x —x$_B$ and increment *A(»).* Local maxima in *A(я)* correspond to instances of the shape in the image.

Note that this is merely an efficient implementation of the convolution of the shape template where edge pixels are unity and others are zero with the corresponding image, i.e.,

$$A(\mathbf{x}) = T(\mathbf{x})^* S(\mathbf{x}) \qquad (15)$$ where E is the binary edge image defined by

$$E(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is an edge pixel} \\ 0 \text{ otherwise} \end{cases}$$

and $T(x)$ is the shape template consisting of ones where x is a boundary point and zeros otherwise, i.e.,

$$T(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is in } B \\ 0 \text{ otherwise} . \end{cases}$$

This result is due to Sklansky.[20]

The Merlin-Farber algorithm is impractical for real image data. In an image with a multitude of edge pixels, there will be many false instances of the desired shape due to coincidental pixel arrangements. Nevertheless, it is the logical precursor to our generalized algorithm.

4.2 *The generalization to arbitrary shapes*

The key to generalizing the Hough algorithm to arbitrary shapes is the use of directional information. Directional information, besides making the algorithm faster, also greatly improves its accuracy. For example, if the directional information is not used in the circle detector, any significant group of edge points with quite different directions which lie on a circle will be detected. This can be appreciated by comparing Figs 2(a) and 2(b).

Consider for a moment the circular boundary detector with a fixed radius r_0. Now for each gradient point x with direction ϕ, we need only increment a single point x+r. For the circle:

$$|r| = r_0 \qquad (16)$$

$$\text{Angle}(\mathbf{r}) = \phi(\mathbf{x}) . \qquad (17)$$

Now suppose we have an arbitrary shape like the one shown in Fig. 6. Extending the idea of the circle detector with fixed radius to this case, for each point x on the boundary with gradient direction ϕ, we increment a point a = x + r. The difference is that now r = a — x which, in general, will vary in magnitude and direction with different boundary points.

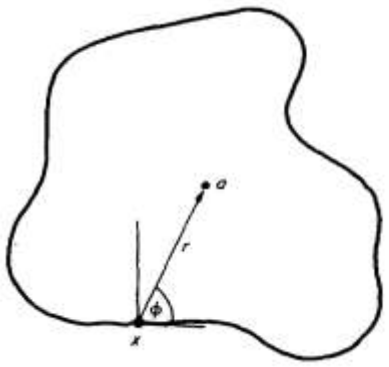


Fig. 6. Geometry for generalized Hough transform.

The fact that r varies in an arbitrary way means that the generalized Hough transform for an arbitrary shape is best represented by a table which we call the Л-table.

4.3 *The R-table*

From the above discussion, we can see that the *R*-table is easily constructed by examining the boundary points of the shape. The construction of the table is accomplished as follows.

Algorithm for constructing an R-table. Choose a reference point y for the shape. For each boundary point x, compute *ф(x)* the gradient direction and r = y —x. Store r as a function of *ф.*

Notice that the mapping the table represents is vector-valued and, in general, an index *ф* may have many values of r. Table 2 shows the form of the Л-table diagrammatically.

The Я-table is used to detect instances of the shape S in an image in the following manner.

Generalized Hough algorithm for single shapes. For each edge pixel x in the image, increment all the corresponding points x + r in the accumulator array *A* where r is a table entry indexed by *ф,* i.e., r(^). Maxima in *A* correspond to possible instances of the shape *S.*

4.4 *Examples*

Some simple shapes are rotation-invariant, that is, the entries in the incrementation table are invariant functions of the gradient direction *ф.* Figure 7(a) shows an example for washers (or bagels). Here there are exactly two entries for each *ф,* one r units in the gradient direction and one *R* units in the direction opposite to the gradient direction. In another case the entries may be a simple function of *ф.* Figure 7(b) shows such an example; hexagons. Irrespective of the orientation of the edge, the reference point locus is on a line of length / parallel to the edge pixel and (3/2)/ units away from it.

Table 2. *R*-table format

i	ϕ_i	R_{ϕ_i}
0	0	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 0\}$
1	$\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = \Delta\phi\}$
2	$2\Delta\phi$	$\{\mathbf{r} \mid \mathbf{a} - \mathbf{r} = \mathbf{x},\ \mathbf{x} \text{ in } B,\ \phi(\mathbf{x}) = 2\Delta\phi\}$

Another example is shown in Fig. 8. Here the points on the boundary of the shape are shown in Fig. 8(a). A reference point is selected and used to construct the *R*-table. Figure 8(b) shows a synthetic image of four different shapes and Fig. 8(c) shows the portion of the accumulator array for this image which has the correct values of orientation and scale. It is readily seen that edge points on the correct shape have incremented the same point in the accumulator array, whereas edge points on the other shapes have incremented disparate points.

4.5 *R-table properties and the general notion of a shape*

Up to this point we have considered shapes of fixed orientation and scale. Thus the accumulator array was two-dimensional in the reference point co-ordinates. To search for shapes of arbitrary orientation *в* and scale 5 we add these two parameters to the shape description. The accumulator array now consists of four dimensions corresponding to the parameters (y, s, *в).* The K-table can also be used to increment this larger dimensional space since different orientations and scales correspond to easily-computed transformations of the table. Additionally, simple transformations to the K-table can also account for figure-ground reversals and changes of reference point.

We denote a particular Л-table for a shape 5 by *Я(ф). R* can be viewed as a multiply-vector-valued function. It is easy to see that simple transformations to this table will allow it to detect scaled or rotated instances of the same shape. For example if the shape is scaled by s and this transformation is denoted by T_s, then

i.e., all the vectors are scaled by s. Also, if the object is rotated by *в* and this transformation is denoted by T_e, then

$$T_\theta[R(\phi)] = \mathrm{Rot}\{R[(\phi-\theta)\bmod 2\pi],\theta\} \qquad (19)$$

i.e., all the indices are incremented by — 0 modulo 2л, the appropriate vectors r are found, and then they are rotated by *в.*

To appreciate that this is true, refer to Fig. 9. In this figure an edge pixel with orientation *ф* may be considered as corresponding to the boundary point x,,, in which case the reference point is y^. Alternatively, the edge pixel may be considered as x_B on a rotated instance of the shape, in which case the reference point is at y_s which can be specified by translating r_A to x_B and rotating it through + Д0.

Figure-ground intensity reversals can also be taken into account via a simple Л-table modification. The indices in the table are changed from *ф* to *(ф + п)mod2n.* Of course

$$T_{fg}\{T_{fg}[R(\phi)]\} = R(\phi)$$

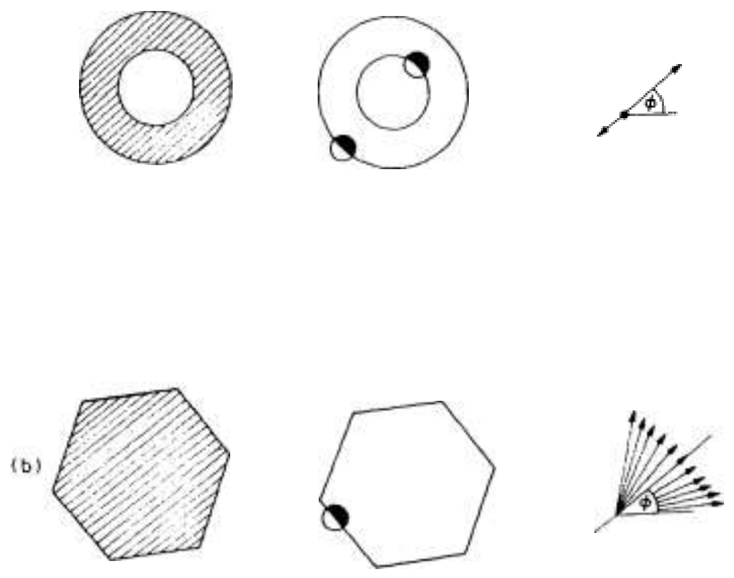


Fig. 7. Simple examples using K-tables; (a) washers; (b) hexagons.

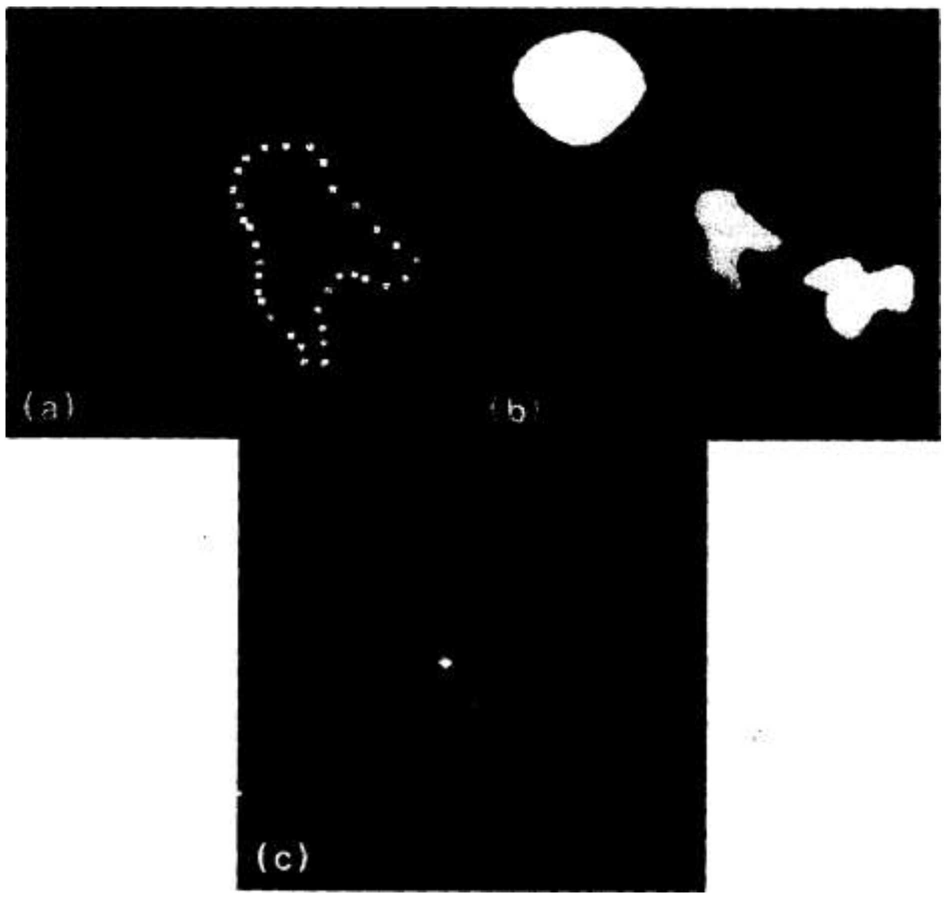


Fig. 8. An example, (a) Points on a shape used to encode K-table. (b) Image containing shape, (c) A plane through the accumulator array *A(x,,* v,, S_0,0_O), where S_0 and θ_0 are appropriate for the shape in the image (S_0 = 64, 9_0 = 0).

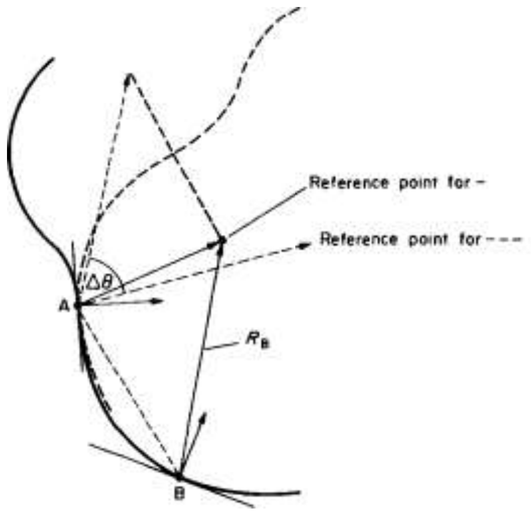


Fig. 9. Construction for visualizing the R-table transformation for a rotation by Д0. Point *A* can be viewed as: (1) on the shape (----), or (2) as point *B* on the shape (-----), rotated by Д0. If (2) is used then the appropriate R is obtained by translating R,, to *A* and rotating it by ДО as shown.

where T_{ft} denotes the figure-ground transformations. Another property which will be useful in describing the composition of generalized Hough transforms is the change of reference point. If we want to choose a new reference point *y'* such that y — y' = r then the modification to the K-table is given by *R(<j>)* + r, i.e. r is added to each vector in the table.

4.6 *Using pairs of edges*

We can also entertain the idea of using pairs of edge pixels to reduce the effort in parameter space. Using the R-table and the properties of the previous section, each edge pixel defines a surface in the four-dimensional accumulator space of a = (y,s,0). Two edge pixels at different orientations describe the same surface rotated by the same amount with respect to *в*. Points where these two surfaces intersect (if any) correspond to possible parameters a for the shape. Thus in a similar manner to Section 3.1, it is theoretically possible to use the two points in image space to reduce the locus in parameter space to a single point. However, the difficulties of finding the intersection points of the two surfaces in parameter space will make this approach unfeasible for most cases.

4.7 *The Hough transform for composite shapes*

Now suppose we have a composite shape S which has two subparts S, and S_2. This shape can be detected by using the K-tables for 5, and S_2 in a remarkably simple fashion. If y, y,, y_2 are the reference points for shapes S, Sj and S_2 respectively, we can compute r, = У-У1 and r_2 = y — y_2- Then the composite generalized Hough transform $И_3(ф)$ is given by

$$K_s(0) = [K_{SI}W) + r.] \, O \, [K_{SI}(0) + r_2] \qquad (20)$$

which means that for each index value </>, r_t is added to K$,(0)> '2 's added to $R_{Sf}(<t>)$, and the union of these sets is stored in $Я_5(ф)$- Equation 20 is very important as it represents a way of composing transforms.

In a similar manner we can define shapes as the difference between tables with common entries, i.e.,

$$Rs — R_s \qquad (21)$$

means the shape *S* defined by S_t with the common entries with S_2 deleted. The intersection operation is defined similarly. The primary use of the union operation is to detect shapes which are composites of simpler shapes. However, the difference operation also serves a useful function. Using it, K-tables which explicitly differentiate between two similar kinds of shapes can be constructed. An example would be differentiating between the washers and hexagons discussed earlier.

4.8 *Building convolution templates*

While equation (20) is one way of composing Hough transforms, it may not be the best way. This is because the choice of reference point can significantly affect the accuracy of the transform. Shapiro"sl617> has shown this, emphasizing analytic forms. This is also graphically shown in Fig. 10. As the reference point becomes distant from the shape, small angular errors in *ф* can produce large errors in the vectors Я(ф).

One solution to this problem is to use the table for each subshape with its own best reference point and to smooth the resultant accumulator array with a composite smoothing template. Recall that for the case of a single shape and isotropic errors (Section 2.2), convolving the accumulator array in this fashion was equivalent to taking account of the errors during the incrementation.

Where *hJy,)* denotes the smoothing template for reference point y, of shape *S*, the composite convolution template is given by

This in practic
introduce error
value and is sui

5. INC

If we use the
lator array by u
lator array are
perimeter of the

nt as it

as the
s, i.e.,

(21)

immon
ition is
union
sites of
эп also
which
nds of
uld be
tagons

Hough
ecause
ectthe
shown
japhi-
comes
</>can

ble
for
and to
i
com-i
seofa
I,
con-n
was
ng the

ite for
: con-

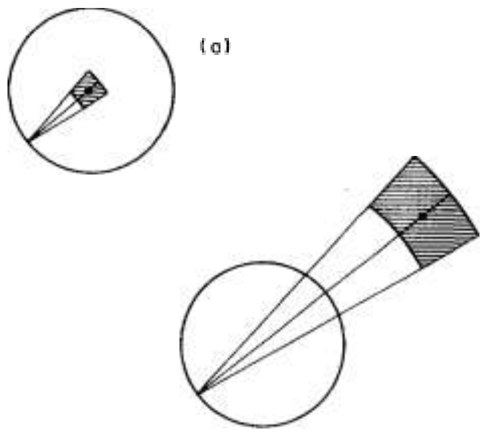


Fig. 10. Effects of changing reference point on errors.

$$H(y) = \sum_{i=1}^{N} h_i(y - y_i). \qquad (22)$$

So finally, we have the following algorithm for the detection of a shape *S* which is the composite of subparts S, ..*S*$_N$.

Generalized Hough algorithm/or composite shapes. 1. For each edge point with direction *ф* and for each value of scale s and orientation *0*, increment the corresponding points x + r in *A* where r is in

адо=гз|г, U/s^) }•

2. Maxima in *A*, = *A*H* correspond to possible instances of the shape *S*. Figure 11 shows a simple example of how templates are combined.

If there are л edge pixels and Af points in the error point spread function template, then the number of additions in the incrementation procedure is *M*. Thus this method might at first seem superior to the convolution method, which requires approximately n^2M additions and multiplications where $M < n^2$, the total number of pixels. However, the following heuristic is available for the convolution since *A* is typicallly very sparse. Compute

/4,(a) only if *A(я)* > 0. (23)

This in practice is very effective, although it may introduce errors if the appropriate index has a zero value and is surrounded by high values.

5. INCREMENTATION STRATEGIES

If we use the strategy of incrementing the accumulator array by unity, then the contents of the accumulator array are approximately proportional to the perimeter of the shape that is detectable in the image.

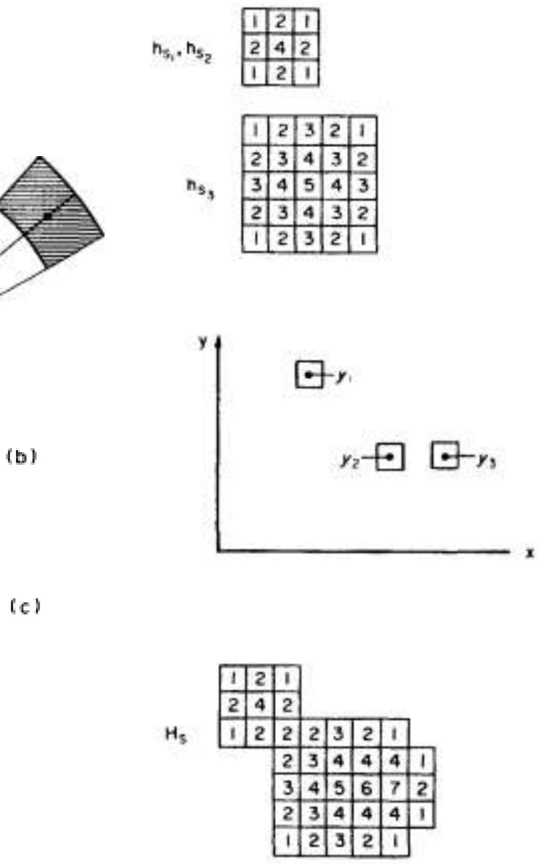


Fig. 11. Example of composite smoothing template construction. (a) Convolution templates for shapes S,, S$_2$, S,. (b) Relationships between reference points y,, *y*$_2$, and y$_3$ in composite shape *S.* (c) Combined smoothing template *Я* as a function of V *h*$_2$, and *h*$_j$ and >',, *y*$_2$, and >'$_3$.

This strategy is biased towards finding shapes where a large portion of the perimeter is detectable. Several different incrementation strategies are available, depending on the different quality of image data. If shorter, very prominent parts of the perimeter are detected, as might be the case in partially occluded objects, then an alternative strategy of incrementing by the gradient modulus value might be more successful, i.e.,

A(л): = A(я) + g(x). (24)

Of course the two strategies can be combined, e.g.,

A(л): = A(ш) + ∂(x) + c (25)

where c is a constant.

Another possibility is the use of local curvature information in the incrementation function. Using this strategy, neighboring edge pixels are examined to calculate approximate curvature, *K.* This requires a more complicated operator than the edge operators we have considered, and complicates the table. Now along with each value of *r* the corresponding values of

curvature must be stored. Then the increment ation

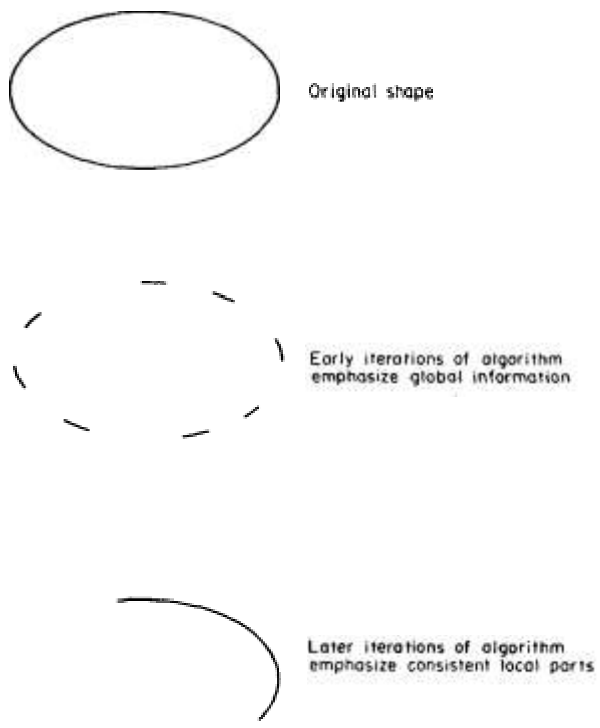

Original shape
Early iterations of algorithm emphasize global information
Later iterations of algorithm emphasize consistent local parts

Fig. 12. Dynamic Hough transform.

weights 'informative' high local curvature edge pixels as follows:

$$A(»): = A(») + K. \tag{26}$$

5.1 *Weighting locally consistent information*

Under certain circumstances we may want to weight local information that is consistent. For example, in searching for the boundary of that object, a connected set of edges conforming to the object may be more important than a set of unconnected edges. Figure 12 shows this example. Figure 12(a) might arise in situations with very noisy data. Figure 12(b) is an example where an object is occluded by another object. Wechsler and Sklansky,[(6)] in the analytic formulation, successfully used the related strategy of increasing the incrementation factor if there were also neighboring edge pixels with the same edge direction. However, we would like to measure local consistency in parameter space.

A simple strategy for handling this case is to explicitly record the reference points for each edge pixel during a first pass. Then on a second pass edge pixels can increment by more than unity if neighboring edge pixels are incrementing the same reference point.

A more complicated strategy is to search for connected curve segments in image space which have compatible parameters. Such an algorithm, based on dynamic programming, is described in Baliard and Sklansky."[4]' The appropriate objective function for a curve segment would be

$$A(x,,Xj,\ldots,x.) = \overset{,,}{£}\, 9(*k) + \overset{,,\_,}{I} <?(x*-x»_{+}i) \tag{27}$$

where

g(\ $_k$) = the gradient magnitude

(28)

and

4(x*.x*+i) = 0 if Иx„) - *Φ*(*$_κ$- i)|mod$_X$ is small and - co otherwise (29)

In the dynamic programming algorithm, at each iteration step we can build longer compatible curves from all the edge points. Thus the incrementation function for a point x would represent the longest compatible curve from that point. (If a longer curve cannot be built at any iteration, we can easily find this out.)

In a parallel implementation of this algorithm the contents of the accumulator array could be made to vary dynamically. Initially the contents would reflect global information, but with successive iterations the contents would be weighted in favor of consistent, local information.

5.2 *More complex strategies*

When searching for a composite object, different parts may have different importance. This is readily accommodated by associating a weight w,- with each table R_{Si} so that each entry in R_s. increments by a factor n>,- instead of unity.

The composite object may be searched for in a sequential manner. Applying the table sequentially could greatly improve the efficiency of the computations by limiting areas for subsequent suitable incrementations. Furthermore, standard methods[21,22] could be used to stop the process once the shape had been located to the desired confidence level.

Even more complex strategies are possible wherein the process is integrated into a larger system. Here contextual information can be used to relegate all the previous operations including (a) building composite templates, (b) choosing weights, (c) choosing application sequences, and (d) adjusting weights in new contexts.