



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

Print/export  
[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)

Languages  
[Français](#)  
[Русский](#)  
[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search

# Generalised Hough transform

From Wikipedia, the free encyclopedia

The **Hough transform** was initially developed to detect analytically defined shapes (e.g., [line](#), [circle](#), [ellipse](#) etc.). In these cases, we have knowledge of the shape and aim to find out its location and orientation in the image. The **Generalised Hough Transform** or GHT, introduced by [Dana H. Ballard](#) in 1981, is the modification of the Hough Transform using the principle of [template matching](#).<sup>[1]</sup> This modification enables the [Hough Transform](#) to be used for not only the detection of an object described with an analytic function. Instead, it can also be used to detect an arbitrary object described with its model.

The problem of finding the object (described with a model) in an image can be solved by finding the model's position in the image. With the Generalised Hough Transform, the problem of finding the model's position is transformed to a problem of finding the transformation's parameter that maps the model into the image. As long as we know the value of the transformation's parameter, the position of the model in the image can be determined.

The original implementation of the GHT uses edge information to define a mapping from orientation of an edge point to a reference point of the shape. In the case of a [binary image](#) where pixels can be either black or white, every black pixel of the image can be a black pixel of the desired pattern thus creating a [locus](#) of reference points in the Hough Space. Every pixel of the image votes for its corresponding reference points. The maximum points of the Hough Space indicate possible reference points of the pattern in the image. This maximum can be found by scanning the Hough Space or by solving a [relaxed set of equations](#), each of them corresponding to a black pixel.<sup>[2]</sup>

## Contents [\[hide\]](#)

- [Earlier Work](#)
- [Theory of Generalised Hough Transform](#)
  - [2.1 Building the R-Table](#)
  - [2.2 Object localization](#)
  - [2.3 Generalization of scale and orientation](#)
  - [2.4 Alternate way using pairs of edges](#)
  - [2.5 Composite shapes](#)
  - [2.6 Spatial Decomposition](#)
- [Implementation](#)<sup>[6]</sup>
- [Advantages and Disadvantages](#)
- [Related work](#)
- [See also](#)
- [References](#)
- [External links](#)

## Earlier Work [\[edit\]](#)

Merlin and Farber<sup>[3]</sup> showed how to use a Hough algorithm when the desired curves could not be described analytically. It was a precursor to Ballard's algorithm but was restricted to [translation](#) and didn't take into account [rotation](#) and [scale](#) changes.<sup>[4]</sup>

The Merlin-Farber algorithm is impractical for real image data as in an image with a large number of edge pixels, there will be many false instances of the desired shape due to similar pixel arrangements.

## Theory of Generalised Hough Transform [\[edit\]](#)

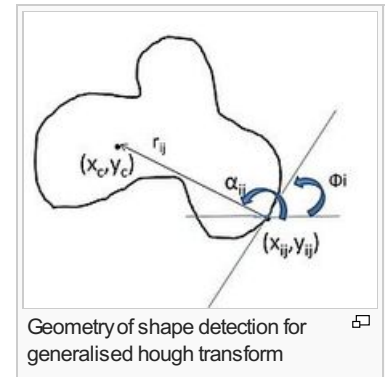
To generalize the Hough algorithm to non-analytic curves, Ballard defines the following parameters for a generalized shape:  $a=\{y,s,\theta\}$  where  $y$  is a reference [origin](#) for the shape,  $\theta$  is its orientation, and  $s = (s_x, s_y)$  describes two [orthogonal](#) scale factors. As in the case of initial Hough Transforms, there is an algorithm for computing the best set of parameters 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 template table called the R table of possible edge pixel orientations. The computation of the additional parameters  $s$  and  $\theta$  is then accomplished

by straightforward transformations to this table. The key to generalizing the Hough algorithm to arbitrary shapes is the use of directional information. Given any shape and a fixed reference point on it, instead of a parametric curve, the information provided by the boundary pixels is stored in the form of the R-table in the transform stage. For every edge point on the test image, the properties of the point are looked up on the R-table and reference point is retrieved and the appropriate cell in a matrix called the Accumulator matrix is incremented. The cell with maximum 'votes' in the Accumulator matrix can be a possible point of existence of fixed reference of the object in the test image.

### Building the R-Table [\[edit\]](#)

Choose a reference point  $y$  for the shape (typically chosen inside the shape). For each boundary point  $x$ , compute  $\phi(x)$ , the [gradient](#) direction and  $r = y - x$  as shown in the image. Store  $r$  as a function of  $\phi$ . Notice that each index of  $\phi$  may have many values of  $r$ . One can either store the co-ordinate differences between the fixed reference and the edge point  $((x_c - x_{ij}), (y_c - y_{ij}))$  or as the radial distance and the angle between them  $(r_{ij}, \alpha_{ij})$ . Having done this for each point, the R-table will fully represent the template object. Also, since the generation phase is invertible, we may use it to localise object occurrences at other places in the image.

| i   | $\phi_i$      | $R_{\phi_i}$  |
|-----|---------------|---|
| 1   | 0             | $(r_{11}, \alpha_{11}) (r_{12}, \alpha_{12}) \dots (r_{1n}, \alpha_{1n})$ |
| 2   | $\Delta\phi$  | $(r_{21}, \alpha_{21}) (r_{22}, \alpha_{22}) \dots (r_{2m}, \alpha_{2m})$ |
| 3   | $2\Delta\phi$ | $(r_{31}, \alpha_{31}) (r_{32}, \alpha_{32}) \dots (r_{3k}, \alpha_{3k})$ |
| ... | ...           | ...   |



### Object localization [\[edit\]](#)

For each edge pixel  $x$  in the image, find the gradient  $\phi$  and increment all the corresponding points  $x+r$  in the accumulator array  $A$  (initialized to a maximum size of the image) where  $r$  is a table entry indexed by  $\phi$ , i.e.,  $r(\phi)$ . These entry points give us each possible position for the reference point. Although some bogus points may be calculated, given that the object exists in the image, a maximum will occur at the reference point. Maxima in  $A$  correspond to possible instances of the shape.

### Generalization of scale and orientation [\[edit\]](#)

For a fixed orientation of shape, the accumulator array was two-dimensional in the reference point co-ordinates. To search for shapes of arbitrary orientation  $\theta$  and scale  $s$ , these two parameters are added to the shape description. The accumulator array now consists of four dimensions corresponding to the parameters  $(y, s, \theta)$ . The R-table can also be used to increment this larger dimensional space since different orientations and scales correspond to easily computed transformations of the table. Denote a particular R-table for a shape  $S$  by  $R(\phi)$ . 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  $T_s[R(\phi)] = sR(\phi)$  i.e., all the vectors are scaled by  $s$ . Also, if the object is rotated by  $\theta$  and this transformation is denoted by  $T_\theta$ , then  $T_\theta[R(\phi)] = Rot\{R[(\phi - \theta) \bmod 2\pi], \theta\}$  i.e., all the indices are incremented by  $-\theta$  modulo  $2\pi$ , the appropriate vectors  $r$  are found, and then they are rotated by  $\theta$ . 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  $\tilde{y}$  such that  $y - \tilde{y} = z$  then the modification to the R-table is given by  $R(\phi) + z$ , i.e.  $z$  is added to each vector in the table.

### Alternate way using pairs of edges [\[edit\]](#)

A pair of edge pixels can be used to reduce the parameter space. Using the R-table and the properties as described above, each edge pixel defines a surface in the four-dimensional accumulator space of  $a = (y, s, \theta)$ . Two edge pixels at different orientations describe the same surface rotated by the same amount with respect to  $\theta$ . If these two surfaces intersect, points where they intersect will correspond to possible parameters  $a$  for the shape. Thus 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.

### Composite shapes [\[edit\]](#)

If the shape  $S$  has a composite structure consisting of subparts  $S_1, S_2 \dots S_N$  and the reference points for the shapes  $S, S_1, S_2 \dots S_N$  are  $y, y_1, y_2 \dots y_n$  respectively, then for a scaling factor  $s$  and orientation  $\theta$ , the

generalized Hough Transform  $R_s(\phi)$  is given by  $R_\phi = T_s \left\{ T_\theta \left[ \bigcup_{k=1}^N R_{s_k}(\phi) \right] \right\}$ . The concern with this transform is that the choice of reference can greatly affect the accuracy. To overcome this, Ballard has suggested smoothing the resultant accumulator with a composite smoothing template. The composite smoothing template  $H(y)$  is given as a composite [convolution](#) of individual smoothing templates of the sub-shapes.  $H(y) = \sum_{i=1}^N h_i(y - y_i)$ . Then the improved Accumulator is given by  $A_s = A^*H$  and the maxima in  $A_s$  corresponds to possible instances of the shape.

### Spatial Decomposition [\[edit\]](#)

Observing that the global Hough Transform can be obtained by the summation of local Hough transforms of disjoint sub-region, Heather and Yang<sup>[5]</sup> proposed a method which involves the [recursive](#) subdivision of the image into sub-images, each with their own parameter space, and organized in a [quadtree](#) structure. It results in improved efficiency in finding endpoints of line segments and improved robustness and reliability in extracting lines in noisy situations, at a slightly increased cost of memory.

### Implementation<sup>[6]</sup> [\[edit\]](#)

$$x = x_c + x' \text{ or } x_c = x - x'$$

$$y = y_c + y' \text{ or } y_c = y - y'$$

$$\cos(\pi - \alpha) = y'/r \text{ or } y' = r\cos(\pi - \alpha) = -r/\sin(\alpha)$$

$$\sin(\pi - \alpha) = x'/r \text{ or } x' = r\sin(\pi - \alpha) = -r/\cos(\alpha)$$

Combining the above equations we have:

$$x_c = x + r\cos(\alpha)$$

$$y_c = y + r\sin(\alpha)$$

#### Constructing the R-Table:

- (0) Convert the sample shape image into an edge image using any edge detecting [edge detecting](#) algorithm like [Canny edge detector](#)
- (1) Pick a reference point (e.g.,  $(x_c, y_c)$ )
- (2) Draw a line from the reference point to the boundary
- (3) Compute  $\phi$
- (4) Store the reference point  $(x_c, y_c)$  as a function of  $\phi$  in  $R(\phi)$  table.

#### Detection:

- (0) Convert the sample shape image into an edge image using any edge detecting algorithm like Canny edge detectors.

- (1) Initialize the Accumulator table:  $A[x_{cmin} \dots x_{cmax}][y_{cmin} \dots y_{cmax}]$
- (2) For each edge point  $(x, y)$ 
  - (2.1) Using the gradient angle  $\phi$ , retrieve from the R-table all the  $(\alpha, r)$  values indexed under  $\phi$ .
  - (2.2) For each  $(\alpha, r)$ , compute the candidate reference points:

$$x_c = x + r\cos(\alpha)$$

$$y_c = y + r\sin(\alpha)$$

- (2.3) Increase counters (voting):

$$++A[[x_c][y_c]]$$

- (3) Possible locations of the object contour are given by local maxima in  $A[x_c][y_c]$ .

If  $A[x_c][y_c] > T$ , then the object contour is located at  $(x_c, y_c)$

#### General Case:

Suppose the object has undergone some rotation  $\Theta$  and uniform scaling  $s$ :

$$(x', y') \rightarrow (x'', y'')$$

$$x'' = (x'\cos(\Theta) - y'\sin(\Theta))s$$

$$y'' = (x'\sin(\Theta) + y'\cos(\Theta))s$$

Replacing  $x'$  by  $x''$  and  $y'$  by  $y''$ :

$$x_c = x - x'' \text{ or } x_c = x - (x'\cos(\Theta) - y'\sin(\Theta))s$$

$$y_c = y - y'' \text{ or } y_c = y - (x'\sin(\Theta) + y'\cos(\Theta))s$$

(1) Initialize the Accumulator table:  $A[x_{cmin} \dots x_{cmax}][y_{cmin} \dots y_{cmax}][q_{min} \dots q_{max}][s_{min} \dots s_{max}]$

(2) For each edge point  $(x, y)$

(2.1) Using its gradient angle  $\phi$ , retrieve all the  $(\alpha, r)$  values from the R-table

(2.2) For each  $(\alpha, r)$ , compute the candidate reference points:

$$x' = r\cos(\alpha)$$

$$y' = r\sin(\alpha)$$

for  $(\Theta = \Theta_{min}; \Theta \leq \Theta_{max}; \Theta++)$

for  $(s = s_{min}; s \leq s_{max}; s++)$

$$x_c = x - (x'\cos(\Theta) - y'\sin(\Theta))s$$

$$y_c = y - (x'\sin(\Theta) + y'\cos(\Theta))s$$

$$++(A[x_c][y_c][\Theta][s])$$

(3) Possible locations of the object contour are given by local maxima in  $A[x_c][y_c][\Theta][s]$

If  $A[x_c][y_c][\Theta][s] > T$ , then the object contour is located at  $(x_c, y_c)$ , has undergone a rotation  $\Theta$ , and has been scaled by  $s$ .

## Advantages and Disadvantages [\[edit\]](#)

### Advantages:

- It is robust to partial or slightly deformed shapes (i.e., robust to recognition under occlusion).
- It is robust to the presence of additional structures in the image.
- It is tolerant to noise.
- It can find multiple occurrences of a shape during the same processing pass.

### Disadvantages:

- It has substantial computational and storage requirements which become acute when object orientation and scale have to be considered.






## Related work [\[edit\]](#)


Ballard suggested using orientation information of the edge decreasing the cost of the computation. Many efficient GHT techniques have been suggested such as the SC-GHT (Using slope and curvature as local properties).<sup>[7]</sup> Davis and Yam<sup>[8]</sup> also suggested an extension of Merlin's work for orientation and scale invariant matching which complement's Ballard's work but does not include Ballard's utilization of edge-slope information and composite structures

## See also [\[edit\]](#)






- [Hough Transform](#)
- [Randomized Hough Transform](#)
- [Radon Transform](#)
- [Template matching](#)
- [Outline of object recognition](#)

## References [\[edit\]](#)

- ↑ D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes", *Pattern Recognition*, Vol.13, No.2, p.111-122, 1981 
- ↑ Jaulin, L.; Bazeille, S. (2013). *Image Shape Extraction using Interval Methods*  (PDF). In *Proceedings of Sysid 2009*, Saint-Malo, France.
- ↑ P. M. Merlin and D. J. Farber, "A parallel mechanism for detecting curves in pictures," *IEEE Trans. Comput.* C24, 96-98 (1975)
- ↑ L. Davis, "Hierarchical Generalized Hough Transforms and Line Segment Based Generalized Hough Transforms" , University of Texas Computer Sciences, Nov 1980
- ↑ J.A. Heather, Xue Dong Yang, "Spatial Decomposition of the Hough Transform" , The 2nd Canadian Conference on Computer and Robot Vision, 2005.
- ↑ Ballard and Brown, [section 4.3.4](#), [Sonka et al., section 5.2.6](#) 
- ↑ A. A. Kassim, T. Tan, K. H. Tan, "A comparative study of efficient generalised Hough transform techniques", *Image and Vision Computing*, Volume 17, Issue 10, Pages 737-748, August 1999

8. <sup>^</sup> L. Davis and S. Yam, "A generalized hough-like transformation for shape recognition"  University of Texas Computer Sciences, TR-134, Feb 1980.

## External links [\[edit\]](#)

- OpenCV implementation of Generalised Hough Transform  
[http://docs.opencv.org/master/d46/classcv\\_1\\_1GeneralizedHoughBallard.html](http://docs.opencv.org/master/d46/classcv_1_1GeneralizedHoughBallard.html) 
- Tutorial and implementation of Generalised Hough transforms <http://www.business-to-technology.com/generalized-hough-transform/default.html> 
- Practical Generalized Hough transform implementation  
[http://www.irit.fr/~Julien.Pinquier/Docs/Hough\\_transform.html](http://www.irit.fr/~Julien.Pinquier/Docs/Hough_transform.html) 
- FPGA implementation of Generalised Hough transforms, IEEE Digital Library  
[http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5382047&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs\\_all.jsp%3Farnumber%3D5382047](http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5382047&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D5382047) 
- MATLAB implementation of Generalised Hough Transform  
<http://www.mathworks.com/matlabcentral/fileexchange/44166-generalized-hough-transform> 

Categories: [Image processing](#)

This page was last modified on 24 July 2015, at 12:16.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

