



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  
العربية  
Català  
Čeština  
Deutsch  
Español  
فارسی  
Français  
한국어  
Italiano  
עברית  
日本語  
Polski  
Português  
Русский  
中文

Edit links

Create account Log in

Article **Talk**

Read **Edit** View history

# Hough transform

From Wikipedia, the free encyclopedia

The **Hough transform** is a [feature extraction](#) technique used in [image analysis](#), [computer vision](#), and [digital image processing](#).<sup>[1]</sup> The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a [parameter space](#), from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of [lines](#) in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by [Richard Duda](#) and [Peter Hart](#) in 1972, who called it a "generalized Hough transform"<sup>[2]</sup> after the related 1962 patent of Paul Hough.<sup>[3][4]</sup> The transform was popularized in the [computer vision](#) community by [Dana H. Ballard](#) through a 1981 journal article titled "[Generalizing the Hough transform to detect arbitrary shapes](#)".

## Contents [hide]

- 1 History
- 2 Theory
- 3 Implementation
- 4 Example
- 5 Variations and extensions
  - 5.1 Using the gradient direction to reduce the number of votes
  - 5.2 Kernel-based Hough transform (KHT)
  - 5.3 3-D Kernel-based Hough transform for plane detection (3DKHT)
  - 5.4 Hough transform of curves, and its generalization for analytical and non-analytical shapes
  - 5.5 Circle detection process
  - 5.6 Detection of 3D objects (Planes and cylinders)
  - 5.7 Using weighted features
  - 5.8 Carefully chosen parameter space
    - 5.8.1 Efficient ellipse detection algorithm
- 6 Limitations
- 7 See also
- 8 References
- 9 External links

## History <sup>[edit]</sup>

It was initially invented for machine analysis of [bubble chamber](#) photographs (Hough, 1959).

The Hough transform was patented as [U.S. Patent 3,069,654](#) in 1962 and assigned to the U.S. Atomic Energy Commission with the name "Method and Means for Recognizing Complex Patterns". This patent uses a slope-intercept parametrization for straight lines, which awkwardly leads to an unbounded transform space since the slope can go to infinity.

The rho-theta parametrization universally used today was first described in

Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM*, Vol. 15, pp. 11–15 (January, 1972),

although it was already standard for the [Radon transform](#) since at least the 1930s.

O'Gorman and Clowes' variation is described in

O'Gorman, Frank; Clowes, MB (1976). "Finding Picture Edges Through Collinearity of Feature Points". *IEEE Trans. Computers* **25** (4): 449–456.

The story of how the modern form of the Hough transform was invented is given in

Hart, P. E., "[How the Hough Transform was Invented](#) (PDF, 268 kB), *IEEE Signal Processing Magazine*, Vol 26, Issue 6, pp 18 - 22 (November, 2009).

### Feature detection

#### Edge detection

[Canny](#) · [Deriche](#) · [Differential](#) · [Sobel](#) · [Prewitt](#) · [Roberts cross](#)

#### Corner detection

[Harris operator](#) · [Shi and Tomasi](#) · [Level curve curvature](#) · [SUSAN](#) · [FAST](#)

#### Blob detection

[Laplacian of Gaussian \(LoG\)](#) · [Difference of Gaussians \(DoG\)](#) · [Determinant of Hessian \(DoH\)](#) · [Maximally stable extremal regions](#) · [PCBR](#)

#### Ridge detection

#### Hough transform

#### Structure tensor

#### Affine invariant feature detection

[Affine shape adaptation](#) · [Harris affine](#) · [Hessian affine](#)

#### Feature description

[SIFT](#) · [SURF](#) · [GLOH](#) · [HOG](#)

#### Scale space

[Scale-space axioms](#) · [Implementation details](#) · [Pyramids](#)

v · t · e

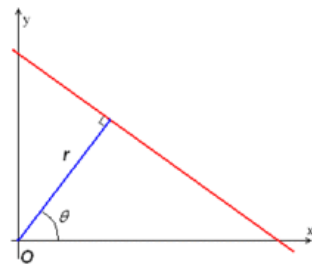
## Theory [\[edit\]](#)

In automated analysis of digital images, a subproblem often arises of detecting simple shapes, such as straight lines, circles or ellipses. In many cases an [edge detector](#) can be used as a pre-processing stage to obtain image points or image pixels that are on the desired curve in the image space. Due to imperfections in either the image data or the edge detector, however, there may be missing points or pixels on the desired curves as well as spatial deviations between the ideal line/circle/ellipse and the noisy edge points as they are obtained from the edge detector. For these reasons, it is often non-trivial to group the extracted edge features to an appropriate set of lines, circles or ellipses. The purpose of the Hough transform is to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects (Shapiro and Stockman, 304).

The simplest case of Hough transform is detecting straight lines. In general, the straight line  $y = mx + b$  can be represented as a point  $(b, m)$  in the parameter space. However, vertical lines pose a problem. They would give rise to unbounded values of the slope parameter  $m$ . Thus, for computational reasons, Duda and Hart<sup>[5]</sup> proposed the use of the [Hesse normal form](#)

$$r = x \cos \theta + y \sin \theta.$$

where  $r$  is the distance from the [origin](#) to the closest point on the straight line, and  $\theta$  (*theta*) is the angle between the  $x$  axis and the line connecting the origin with that closest point.



It is therefore possible to associate with each line of the image a pair  $(r, \theta)$ . The  $(r, \theta)$  plane is sometimes referred to as *Hough space* for the set of straight lines in two dimensions. This representation makes the Hough transform conceptually very close to the two-dimensional [Radon transform](#). (They can be seen as different ways of looking at the same transform.<sup>[6]</sup>)

Given a *single point* in the plane, then the set of *all* straight lines going through that point corresponds to a [sinusoidal curve](#) in the  $(r, \theta)$  plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at the  $(r, \theta)$

for that line. Thus, the problem of detecting [collinear points](#) can be converted to the problem of finding [concurrent curves](#).<sup>[7]</sup>

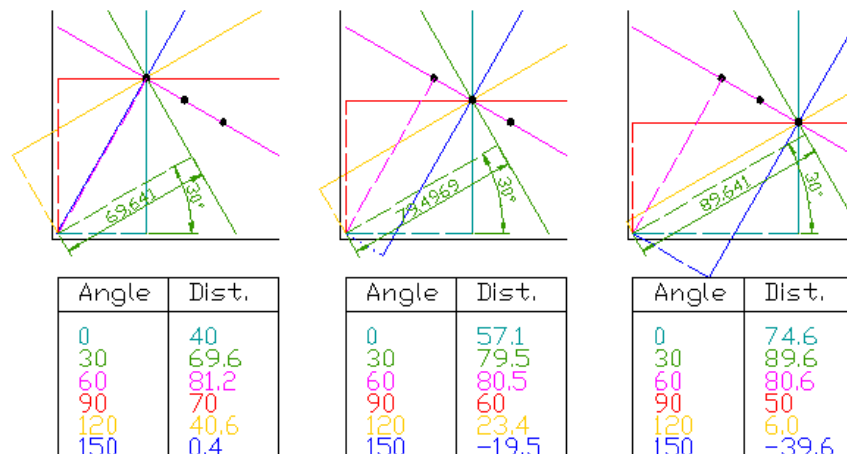
## Implementation [\[edit\]](#)

The linear Hough transform [algorithm](#) uses a two-dimensional array, called an accumulator, to detect the existence of a line described by  $r = x \cos \theta + y \sin \theta$ . The [dimension](#) of the accumulator equals the number of unknown parameters, i.e., two, considering quantized values of  $r$  and  $\theta$  in the pair  $(r, \theta)$ . For each pixel at  $(x, y)$  and its neighborhood, the Hough transform algorithm determines if there is enough evidence of a straight line at that pixel. If so, it will calculate the parameters  $(r, \theta)$  of that line, and then look for the accumulator's bin that the parameters fall into, and increment the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off. (Shapiro and Stockman, 304) The simplest way of finding these *peaks* is by applying some form of threshold, but other techniques may yield better results in different circumstances - determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often necessary, in the next step, to find which parts of the image match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines.

The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator—one dimension of this matrix is the quantized angle  $\theta$  and the other dimension is the quantized distance  $r$ . Each element of the matrix has a value equal to the number of points or pixels that are positioned on the line represented by quantized parameters  $(r, \theta)$ . So the element with the highest value indicates the straight line that is most represented in the input image.<sup>[8]</sup>

## Example [\[edit\]](#)

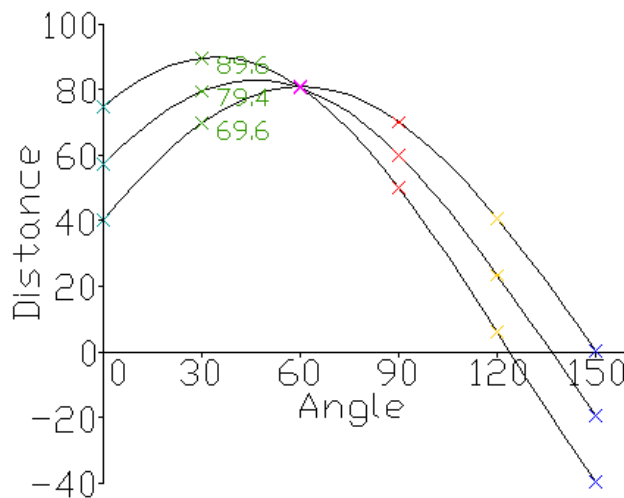
Consider three data points, shown here as black dots.



- For each data point, a number of lines are plotted going through it, all at different angles. These are shown here as solid

lines.

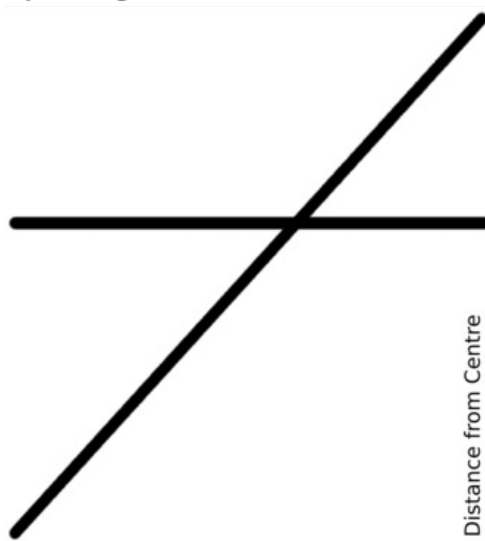
- For each solid line a line is plotted which is [perpendicular](#) to it and which intersects the [origin](#). These are shown as dashed lines.
- The length (i.e. perpendicular distance to the origin) and angle of each dashed line is measured. In the diagram above, the results are shown in tables.
- This is repeated for each data point.
- A graph of the line lengths for each angle, known as a Hough space graph, is then created.



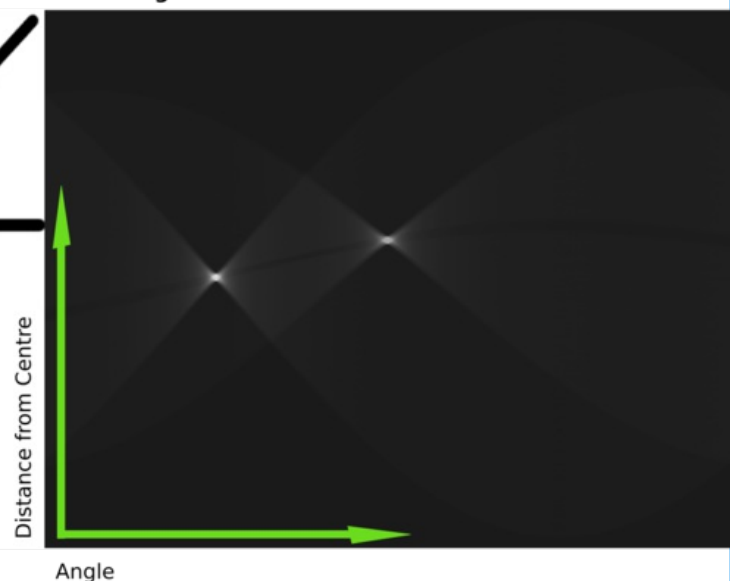
The point where the curves intersect gives a distance and angle. This distance and angle indicate the line which intersects the points being tested. In the graph shown the lines intersect at the pink point; this corresponds to the solid pink line in the diagrams above, which passes through all three points.

The following is a different example showing the results of a Hough transform on a raster image containing two thick lines.

**Input Image**



**Rendering of Transform Results**



The results of this transform were stored in a matrix. Cell value represents the number of curves through any point. Higher cell values are rendered brighter. The two distinctly bright spots are the Hough parameters of the two lines. From these spots' positions, angle and distance from image center of the two lines in the input image can be determined.

## Variations and extensions [\[edit\]](#)

### Using the gradient direction to reduce the number of votes [\[edit\]](#)

An improvement suggested by O'Gorman and Clowes can be used to detect lines if one takes into account that the local [gradient](#) of the image intensity will necessarily be orthogonal to the edge. Since [edge detection](#) generally involves computing the intensity [gradient](#) magnitude, the gradient direction is often found as a side effect. If a given point of coordinates  $(x,y)$  happens to indeed be on a line, then the local direction of the gradient gives the  $\theta$  parameter corresponding to said line, and the  $r$  parameter is then immediately obtained. (Shapiro and Stockman, 305) The gradient direction can be estimated to within  $20^\circ$ , which shortens the sinusoid trace from the full  $180^\circ$  to roughly  $45^\circ$ . This reduces the computation time and has the interesting effect of reducing the number of useless votes, thus enhancing the visibility of the spikes corresponding to real lines in the image.

### Kernel-based Hough transform (KHT) [\[edit\]](#)

Fernandes and Oliveira [9] suggested an improved voting scheme for the Hough transform that allows a software implementation to achieve real-time performance even on relatively large images (e.g., 1280×960). The Kernel-based Hough transform uses the same  $(r, \theta)$  parameterization proposed by Duda and Hart but operates on clusters of approximately collinear pixels. For each cluster, votes are cast using an oriented elliptical-Gaussian kernel that models the uncertainty associated with the best-fitting line with respect to the corresponding cluster. The approach not only significantly improves the performance of the voting scheme, but also produces a much cleaner accumulator and makes the transform more robust to the detection of spurious lines.

### 3-D Kernel-based Hough transform for plane detection (3DKHT) [edit]

Limberger and Oliveira [10] suggested a deterministic technique for plane detection in unorganized point clouds whose cost is  $n \log(n)$  in the number of samples, achieving real-time performance for relatively large datasets (up to  $10^5$  points on a 3.4 GHz CPU). It is based on a fast Hough-transform voting strategy for planar regions, inspired by the Kernel-based Hough transform (KHT). This 3D Kernel-based Hough transform (3DKHT) uses a fast and robust algorithm to segment clusters of approximately co-planar samples, and casts votes for individual clusters (instead of for individual samples) on a  $(\theta, \phi, \rho)$  spherical accumulator using a trivariate Gaussian kernel. The approach is several orders of magnitude faster than existing (non-deterministic) techniques for plane detection in point clouds, such as RHT and RANSAC, and scales better with the size of the datasets. It can be used with any application that requires fast detection of planar features on large datasets.

### Hough transform of curves, and its generalization for analytical and non-analytical shapes [edit]

Although the version of the transform described above applies only to finding straight lines, a similar transform can be used for finding any shape which can be represented by a set of parameters. A circle, for instance, can be transformed into a set of three parameters, representing its center and radius, so that the Hough space becomes three dimensional. Arbitrary ellipses and curves can also be found this way, as can any shape easily expressed as a set of parameters.

The generalization of the Hough transform for detecting analytical shapes in spaces having any dimensionality was proposed by Fernandes and Oliveira. [11] In contrast to other Hough transform-based approaches for analytical shapes, Fernandes' technique does not depend on the shape one wants to detect nor on the input data type. The detection can be driven to a type of analytical shape by changing the assumed model of geometry where data have been encoded (e.g., [euclidean space](#), [projective space](#), [conformal geometry](#), and so on), while the proposed formulation remains unchanged. Also, it guarantees that the intended shapes are represented with the smallest possible number of parameters, and it allows the concurrent detection of different kinds of shapes that best fit an input set of entries with different dimensionalities and different geometric definitions (e.g., the concurrent detection of planes and spheres that best fit a set of points, straight lines and circles).

For more complicated shapes in the plane (i.e., shapes that cannot be represented analytically in some 2D space), the [Generalised Hough transform](#) [12] is used, which allows a feature to vote for a particular position, orientation and/or scaling of the shape using a predefined look-up table.

### Circle detection process [edit]

The process of identifying possible [circular objects in Hough space](#) is relatively simple,

- First we create our accumulator space which is made up of a cell for each pixel, initially each of these will be set to 0.
- For each(edge point in image(i, j)): Increment all cells which according to the equation of a circle  $(i-a)^2 + (j-b)^2 = r^2$  could be the center of a circle, these cells are represented by the letter 'a' in the equation.
- For all possible value of a found in the previous step, find all possible values of b which satisfy the equation.
- Search for the local maxima cells, these are any cells whose value is greater than every other cell in its neighbourhood.

These cells are the one with the highest probability of being the location of the circle(s) we are trying to locate.

Note that in most problems we will know the radius of the circle we are trying to locate beforehand, however if this is not the case we can use a three-dimensional accumulator space, this is much more computationally expensive. This method can also detect circles that are partially outside of the accumulator space if enough of its area is still present within it.

### Detection of 3D objects (Planes and cylinders) [edit]

Hough transform can also be used for the detection of 3D objects in range data or 3D [point clouds](#). The extension of classical Hough transform for plane detection is quite straightforward. A plane is represented by its explicit equation  $z = a_x x + a_y y + d$  for which we can use a 3D Hough space corresponding to  $a_x$ ,  $a_y$  and  $d$ . This extension suffers from the same problems as its 2D counterpart i.e., near horizontal planes can be reliably detected, while the performance deteriorates as planar direction becomes vertical (big values of  $a_x$  and  $a_y$  amplify the noise in the data). This formulation of the plane has been used for the detection of planes in the [point clouds](#) acquired from airborne laser scanning [13] and works very well because in that domain all planes are nearly horizontal.

For generalized plane detection using Hough transform, the plane can be parametrized by its normal vector  $\mathbf{n}$  (using spherical coordinates) and its distance from the origin  $\rho$  resulting in a three dimensional Hough space. This results in each point in the input data voting for a sinusoidal surface in the Hough space. The intersection of these sinusoidal surfaces indicates presence of a plane. [14] A more general approach for more than 3 dimensions requires search heuristics to remain feasible. [15]

Hough transform has also been used to find cylindrical objects in [point clouds](#) using a two step approach. The first step finds the orientation of the cylinder and the second step finds the position and radius.<sup>[16]</sup>

### Using weighted features [\[edit\]](#)

One common variation detail. That is, finding the bins with the highest count in one stage can be used to constrain the range of values searched in the next.

### Carefully chosen parameter space [\[edit\]](#)

A high-dimensional parameter space for the Hough transform is not only slow, but if implemented without forethought can easily overrun the available memory. Even if the programming environment allows the allocation of an array larger than the available memory space through virtual memory, the number of page swaps required for this will be very demanding because the accumulator array is used in a randomly accessed fashion, rarely stopping in contiguous memory as it skips from index to index.

Consider the task of finding ellipses in an 800x600 image. Assuming that the radii of the ellipses are oriented along principal axes, the parameter space is four-dimensional. (x,y) defines the center of the ellipse, and a and b denote the two radii. Allowing the center to be anywhere in the image, adds the constraint  $0 < x < 800$  and  $0 < y < 600$ . If the radii are given the same values as constraints, what is left is a sparsely filled accumulator array of more than 230 billion values.

A program thus conceived is unlikely to be allowed to allocate sufficient memory. This doesn't mean that the problem can't be solved, but only that new ways to constrain the size of the accumulator array are to be found, which makes it feasible. For instance:

1. If it is reasonable to assume that the ellipses are each contained entirely within the image, the range of the radii can be reduced. The largest the radii can be is if the center of the ellipse is in the center of the image, allowing the edges of the ellipse to stretch to the edges. In this extreme case, the radii can only each be half the magnitude of the image size oriented in the same direction. Reducing the range of a and b in this fashion reduces the accumulator array to 57 billion values.
2. Trade accuracy for space in the estimation of the center: If the center is predicted to be off by 3 on both the x and y axis this reduces the size of the accumulator array to about 6 billion values.
3. Trade accuracy for space in the estimation of the radii: If the radii are estimated to each be off by 5 further reduction of the size of the accumulator array occurs, by about 256 million values.
4. Crop the image to areas of interest. This is image dependent, and therefore unpredictable, but imagine a case where all of the edges of interest in an image are in the upper left quadrant of that image. The accumulator array can be reduced even further in this case by constraining all 4 parameters by a factor of 2, for a total reduction factor of 16.

By applying just the first three of these constraints to the example stated about, the size of the accumulator array is reduced by almost a factor of 1000, bringing it down to a size that is much more likely to fit within a modern computer's memory.

### Efficient ellipse detection algorithm [\[edit\]](#)

Yonghong Xie and Qiang Ji give an efficient way of implementing the Hough transform for ellipse detection by overcoming the memory issues.<sup>[17]</sup> As discussed in the algorithm (on page 2 of the paper), this approach uses only a one-dimensional accumulator (for the minor axis) in order to detect ellipses in the image. The complexity is  $O(N^3)$  in the number of non-zero points in the image.

## Limitations [\[edit\]](#)

The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin.<sup>[18]</sup>

Also, when the number of parameters is large (that is, when we are using the Hough transform with typically more than three parameters), the average number of votes cast in a single bin is very low, and those bins corresponding to a real figure in the image do not necessarily appear to have a much higher number of votes than their neighbors. The complexity increases at a rate of  $O(A^{m-2})$  with each additional parameter, where  $A$  is the size of the image space and  $m$  is the number of parameters. (Shapiro and Stockman, 310) Thus, the Hough transform must be used with great care to detect anything other than lines or circles.



Finally, much of the efficiency of the Hough transform is dependent on the quality of the input data: the edges must be detected well for the Hough transform to be efficient. Use of the Hough transform on noisy images is a very delicate matter and generally, a denoising stage must be used before. In the case where the image is corrupted by speckle, as is the case in radar images, the [Radon transform](#) is sometimes preferred to detect lines, because it attenuates the noise through summation.

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

- [Generalised Hough transform](#)
- [Randomized Hough transform](#)
- [Radon transform](#)

- [Fourier transform](#)

## References [[edit](#)]

- ↑ Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. 2001
- ↑ Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM*, Vol. 15, pp. 11–15 (January, 1972)
- ↑ Hough, P.V.C. *Method and means for recognizing complex patterns*, U.S. Patent 3,069,654, Dec. 18, 1962
- ↑ P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959
- ↑ Richard O. Duda and Peter E. Hart (April 1971). "Use of the Hough Transformation to Detect Lines and Curves in Pictures"  (PDF). *Artificial Intelligence Center* (SRI International).
- ↑ CiteSeerX — A short introduction to the Radon and Hough transforms and how they relate to each other [↗](#)
- ↑ "Hough Transform" [↗](#).
- ↑ Jensen, Jeppe. "Hough Transform for Straight Lines"  (PDF). Retrieved 16 December 2011.
- ↑ Fernandes, L.A.F.; Oliveira, M.M. (2008). "Real-time line detection through an improved Hough transform voting scheme". *Pattern Recognition* **41** (1): 299–314. doi:10.1016/j.patcog.2007.04.003 [↗](#).
- ↑ Limberger, F.A.; Oliveira, M.M. (2015). "Real-Time Detection of Planar Regions in Unorganized Point Clouds". *Pattern Recognition* **48** (6): 2043–2053. doi:10.1016/j.patcog.2014.12.020 [↗](#).
- ↑ Fernandes, L.A.F.; Oliveira, M.M. (2012). "A general framework for subspace detection in unordered multidimensional data". *Pattern Recognition* **45** (9): 3566–3579. doi:10.1016/j.patcog.2012.02.033 [↗](#).
- ↑ Ballard, D.H. (1981). "Generalizing the Houghtransform to detectarbitraryshapes". *Pattern Recognition* **13** (2): 111–122. doi:10.1016/0031-3203(81)90009-1 [↗](#).
- ↑ Vosselman, G., Dijkman, S: "3D Building Model Reconstruction from Point Clouds and Ground Plans", International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol 34, part 3/W4, October 22–24, 2001, Annapolis, MA, USA, pp.37- 44.
- ↑ Tahir Rabbani: "Automatic reconstruction of industrial installations - Using point clouds and images", page 43-44, Publications on Geodesy 62, Delft, 2006. ISBN 978-90-6132-297-9 <http://www.ncg.knaw.nl/Publicaties/Geodesy/62Rabbani.html> [↗](#)
- ↑ Achtert, Elke; Böhm, Christian; David, Jörn; Kröger, Peer; Zimek, Arthur (2008). "Global Correlation Clustering Based on the Hough Transform". *Statistical Analysis and Data Mining* **1** (3): 111–127. doi:10.1002/sam.10012 [↗](#).
- ↑ Tahir Rabbani and Frank van den Heuvel, "Efficient hough transform for automatic detection of cylinders in point clouds" in Proceedings of the 11th Annual Conference of the Advanced School for Computing and Imaging (ASCI '05), The Netherlands, June 2005.
- ↑ Yonghong Xie; Qiang Ji (2002). "A new efficient ellipse detection method" **2**. p. 957. doi:10.1109/ICPR.2002.1048464 [↗](#).
- ↑ "Image Transforms - Hough Transform" [↗](#). Homepages.inf.ed.ac.uk. Retrieved 2009-08-17.

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

- [hough\\_transform.cpp](#) [↗](#) - C++ code - example of Clmg library ([open source](#) library, [C++](#) source code, [Grayscale](#) images)
- [Interactive Demonstration on the Basics of the Hough Transform](#) [↗](#)
- <http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/Hough.html> [↗](#) - [Java Applet](#) + Source for learning the Hough transformation in slope-intercept form
- <http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/HNF.html> [↗](#) - [Java Applet](#) + Source for learning the Hough-Transformation in normal form
- <http://www.sydlogan.com/deskew.html> [↗](#) - Deskew images using Hough transform ([Grayscale](#) images, [C++](#) source code)
- <http://imaging.gmse.net/articledeskew.html> [↗](#) - Deskew images using Hough transform ([Visual Basic](#) source code)
- <http://www.mitov.com/products/visionlab> [↗](#) - [Delphi](#), [C++](#) and [.NET](#) free for educational purposes library containing Line, Circle and Line segment Hough transform components.
- Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., 2007a. Hough-transform and extended RANSAC algorithms for automatic detection of 3d building roof planes from Lidar data.  ISPRS Proceedings. Workshop Laser scanning. Espoo, Finland, September 12–14, 2007.
- [Into](#) [↗](#) contains open source implementations of linear and circular Hough transform in C++
- <http://www.vision.ime.usp.br/~edelgado/defesa/code/hough.html> [↗](#) Hough-transform for Ellipse detection, implemented in C.
- [scikit-image](#) [↗](#) Hough-transform for line, circle and ellipse, implemented in Python.
- [\[1\]](#) [↗](#) Hough transform based on wavelet filtering, to detect a circle of a particular radius. (Matlab code.)
- [Hough transform for lines using MATLAB](#) [↗](#)
- [Hough transform for circles and ellipses in MATLAB](#) [↗](#)
- [KHT](#) [↗](#) - C++ source code.
- [3DKHT](#) [↗](#) - C++ source code and datasets.

Categories: [Feature detection \(computer vision\)](#)

This page was last modified on 4 September 2015, at 07:36.

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.

