Hough transform

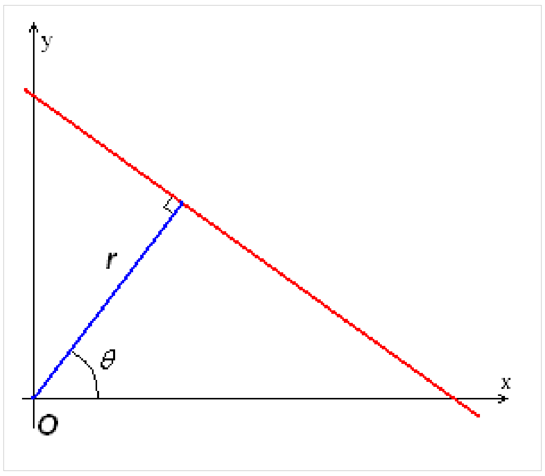
The **Hough transform** (/ˈhʌf/) is a feature extraction technique used in image analysis, computer vision, and digital image processing.[1] The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried 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"[2] after the related 1962 patent of Paul Hough.[3] 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".

Theory

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 the linear transform for detecting straight lines. In the image space, the straight line can be described as $y = mx + b$ and can be graphically plotted for each pair of image points (x, y). In the Hough transform, a main idea is to consider the characteristics of the straight line not as image points (x_1, y_1), (x_2, y_2), etc., but instead, in terms of its parameters, i.e., the slope parameter m and the intercept parameter b. Based on that fact, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, one faces the problem that vertical lines give rise to unbounded values of the parameters m and b. For computational reasons, it is therefore better to use a different pair of parameters, denoted r and θ (*theta*), for the lines in the Hough transform. These are the Polar Coordinates.



The parameter r represents the distance between the line and the origin, while θ is the angle of the vector from the origin to this closest point (see Coordinates). Using this parameterization, the equation of the line can be written as[4]

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

which can be rearranged to $r = x\cos\theta + y\sin\theta$ (Shapiro and Stockman, 304).

It is therefore possible to associate with each line of the image a pair (r,θ) which is unique if $\theta \in [0,\pi)$ and $r \in \mathbf{R}$, or if $\theta \in [0,2\pi)$ and r 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.[5])

For an arbitrary point on the image plane with coordinates, e.g., (x_0, y_0), the lines that go through it are

$$r(\theta) = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta \ ,$$

where r (the distance between the line and the origin) is determined by θ.

This corresponds to a sinusoidal curve in the (r,θ) plane, which is unique to that point. If the curves corresponding to two points are superimposed, the location (in the *Hough space*) where they cross corresponds to a line (in the original image space) that passes through both points. More generally, a set of points that form a straight line will produce sinusoids which cross at the parameters for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves.[6]

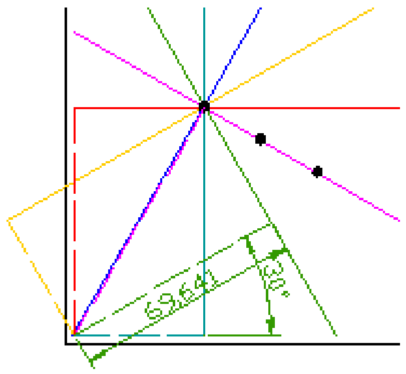
Implementation

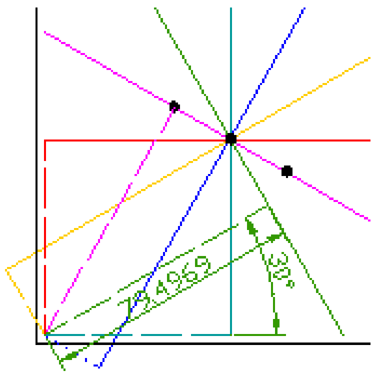
The Hough transform algorithm uses an array, called an accumulator, to detect the existence of a line y = mx + b. The dimension of the accumulator is equal to the number of unknown parameters of the Hough transform problem. For example, the linear Hough transform problem has two unknown parameters: the pair (m,b) or the pair (r,θ). The two dimensions of the accumulator array would correspond to quantized values for (r,θ). For each pixel and its neighborhood, the Hough transform algorithm determines if there is enough evidence of an edge at that pixel. If so, it will calculate the parameters of that line, and then look for the accumulator's bin that the parameters fall into, and increase 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 different 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 next necessary 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.

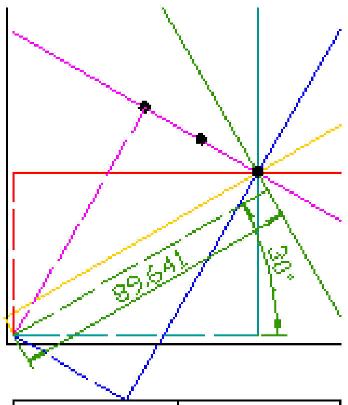
The result of the Hough transform is stored in a matrix that often is called an accumulator. One dimension of this matrix are the angles θ and the other dimension are the distances r, and each element has a value telling how many points/pixels are positioned on the line with parameters (r,θ). So the element with the highest value tells what line that is most represented in the input image.[7]

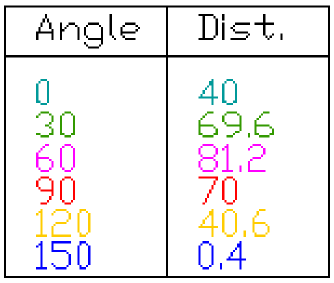
Example

Consider three data points, shown here as black dots.

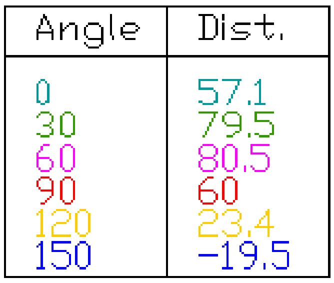




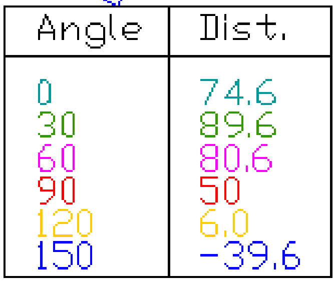




Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

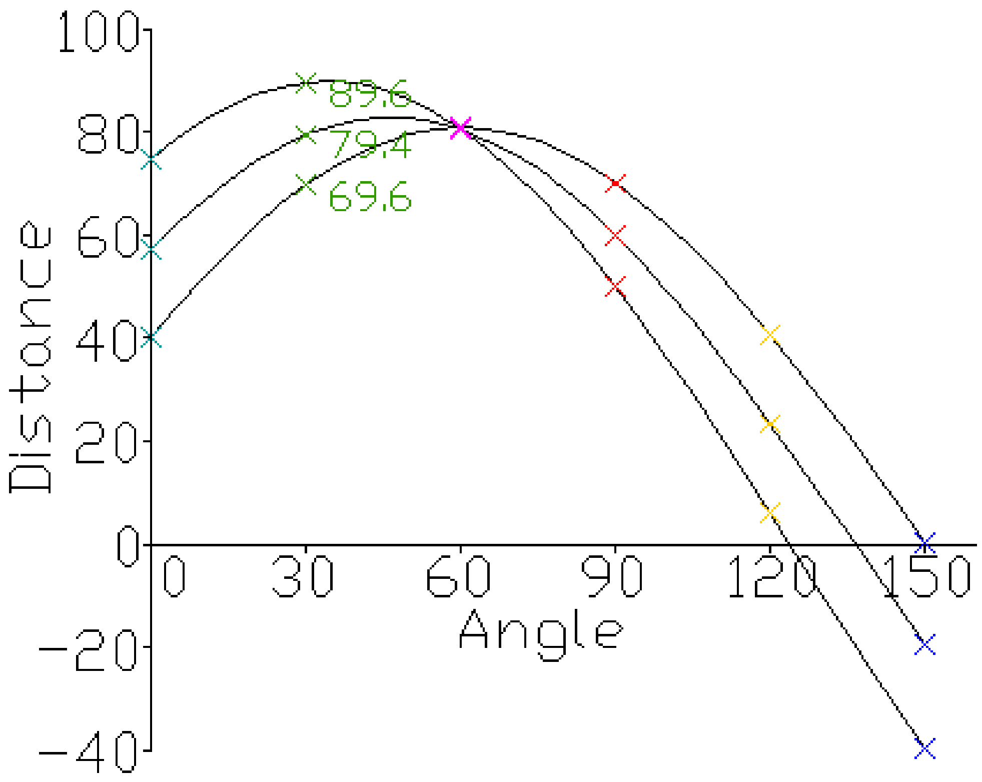


Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5



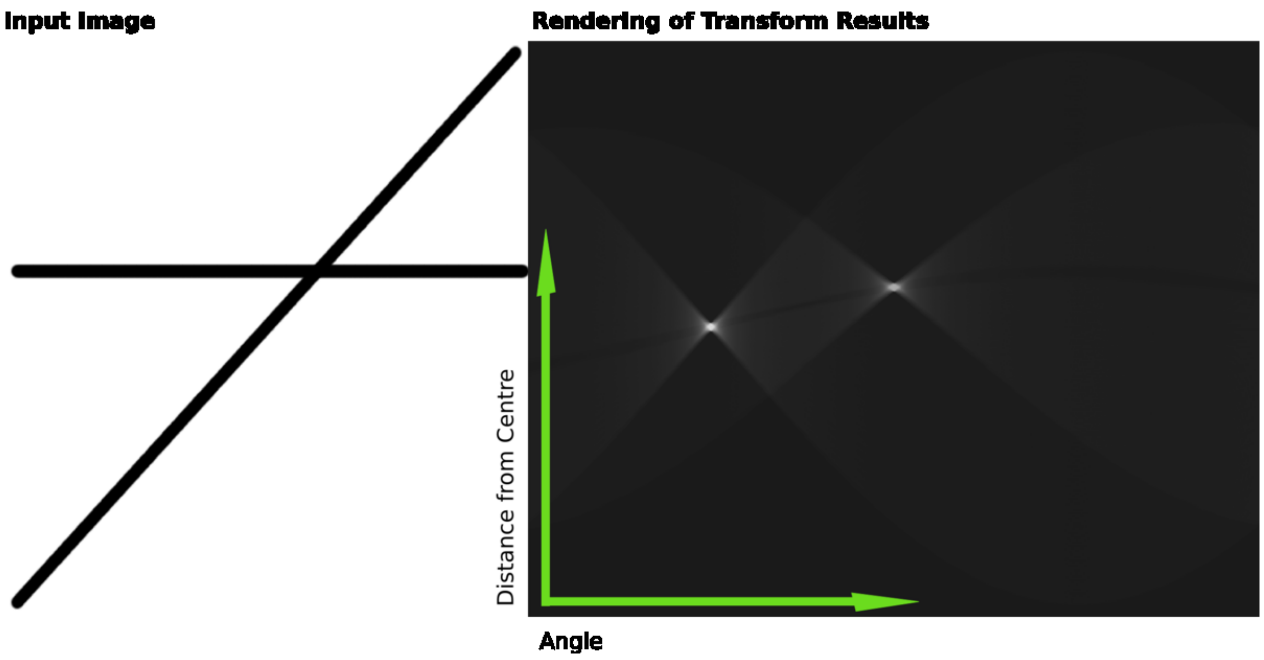
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

- 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.



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

Using the gradient direction to reduce the number of votes

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 θ 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°, which shortens the sinusoid trace from the full 180° to roughly 45°. 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

Fernandes and Oliveira [8] 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, θ) 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.

Hough transform of curves, and Generalised Hough transform

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. For more complicated shapes, the Generalised Hough transform 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.

Detection of 3D objects (Planes and cylinders)

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 straight forward. 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 counter part 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 [9] 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 n (using spherical coordinates) and its distance from the origin ρ 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.[10] A more general approach for more than 3 dimensions requires search heuristics to remain feasible.[11]

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.[12]

Using weighted features

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

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. Matlab has functions aimed specifically for sparse matrices, but they only handle two dimensional matrices, not four dimensional.

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

Yonghong Xie and Qiang Ji [13] give an efficient way of implementing Hough Transform for Ellipse detection by overcoming the memory issues. As discussed in the algorithm (on Page 2 of the paper), this approach uses only a one dimensional accumulator (for 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

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.[14]

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 $\mathcal{O}\left(A^{m-2}\right)$ with each additional parameter, where A is 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.

History

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

The Hough transform was patented as U.S. Patent 3069654 [15] 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

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

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", *IEEE Signal Processing Magazine, Vol 26, Issue 6*, pp 18 - 22 (November, 2009) .

References

[1] Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. 2001

[2] 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)

[3] P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

[4] "Use of the Hough Transformation to Detect Lines and Curves in Pictures" (http://www.ai.sri.com/pubs/files/tn036-duda71.pdf). .

[5] CiteSeerX — A short introduction to the Radon and Hough transforms and how they relate to each other (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.9419)

[6] "Hough Transform" (http://planetmath.org/encyclopedia/HoughTransform.html). .

[7] Jensen, Jeppe. "Hough Transform for Straight Lines" (http://www.cvmt.dk/education/teaching/e07/MED3/IP/hough_lines.pdf). . Retrieved 16 December 2011.

[8] Fernandes, L.A.F. and Oliveira, M.M., "Real-time line detection through an improved Hough transform voting scheme," *Pattern Recognition, Elsevier, Volume 41, Issue 1*, pp. 299–314 (January, 2008) (http://dx.doi.org/10.1016/j.patcog.2007.04.003).

[9] 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.

[10] 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

[11] Elke Achtert, Christian Böhm, Jörn David, Peer Kröger, Arthur Zimek: "Global Correlation Clustering Based on the Hough Transform", Statistical Analysis and Data Mining, vol 1(3), pp. 111-127, 2008. http://dx.doi.org/10.1002/sam.10012

[12] 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.

[13] Yonghong Xie; Qiang Ji (2002). *A new efficient ellipse detection method*. **2**. pp. 957. doi:10.1109/ICPR.2002.1048464.

[14] "Image Transforms - Hough Transform" (http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm). Homepages.inf.ed.ac.uk. . Retrieved 2009-08-17.

[15] http://www.google.com/patents?vid=3069654

External links

- hough_transform.cpp (http://cimg.sourceforge.net/screenshots.shtml) - C++ code - example of CImg library (open source library, C++ source code, Grayscale images)
- 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. (http://foto.hut.fi/seura/julkaisut/pjf/pjf_e/2008/Tarsha-Kurdi_et_al_2008_PJF.pdf) ISPRS Proceedings. Workshop Laser scanning. Espoo, Finland, September 12–14, 2007.
- Into (http://intopii.com/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.

Article Sources and Contributors

Image Sources, Licenses and Contributors

License