Circle Detection Notes

Steven Rankine

Contents

1 Edge detection **1**

1.1 Motivations . . . 1
1.2 Edge properties . . . 1
1.3 A simple edge model . . . 1
1.4 Why edge detection is a non-trivial task . . . 2
1.5 Approaches . . . 2
1.5.1 Canny edge detection . . . 2
1.5.2 Other first-order methods . . . 3
1.5.3 Thresholding and linking . . . 3
1.5.4 Edge thinning . . . 3
1.5.5 Second-order approaches to edge detection . . . 4
1.5.6 Phase congruency-based edge detection . . . 5
1.5.7 Physics-inspired edge detection . . . 5
1.6 See also . . . 5
1.7 References . . . 5
1.8 Further reading . . . 6

2 Dilation (morphology) **7**

2.1 Binary operator . . . 7
2.1.1 Properties of binary dilation . . . 7
2.2 Grayscale dilation . . . 7
2.2.1 Flat structuring functions . . . 8
2.3 Dilation on complete lattices . . . 8
2.4 See also . . . 8
2.5 Bibliography . . . 8

3 Blob detection **9**

3.1 The Laplacian of Gaussian . . . 9
3.2 The difference of Gaussians approach . . . 10
3.3 The determinant of the Hessian . . . 10
3.4 The hybrid Laplacian and determinant of the Hessian operator (Hessian-Laplace) . . . 10
3.5 Affine-adapted differential blob detectors . . . 11
3.6 Grey-level blobs, grey-level blob trees and scale-space blobs . . . 11

3.6.1 Lindeberg's watershed-based grey-level blob detection algorithm . . . 11
3.7 Maximally stable extremum regions (MSER) . . . 12
3.8 See also . . . 12
3.9 References . . . 12

4 Hough transform **14**
4.1 History . . . 14
4.2 Theory . . . 14
4.3 Implementation . . . 15
4.4 Example . . . 15
4.5 Variations and extensions . . . 16
4.5.1 Using the gradient direction to reduce the number of votes . . . 16
4.5.2 Kernel-based Hough transform (KHT) . . . 17
4.5.3 3-D Kernel-based Hough transform for plane detection (3DKHT) . . . 17
4.5.4 Hough transform of curves, and its generalization for analytical and non-analytical shapes . . . 17
4.5.5 Circle detection process . . . 17
4.5.6 Detection of 3D objects (Planes and cylinders) . . . 18
4.5.7 Using weighted features . . . 18
4.5.8 Carefully chosen parameter space . . . 18
4.6 Limitations . . . 19
4.7 See also . . . 19
4.8 References . . . 19
4.9 External links . . . 20

5 Circle Hough Transform **21**
5.1 Theory . . . 21
5.1.1 Find parameters with Known radius R . . . 21
5.1.2 Multiple circles with known radius R . . . 21
5.1.3 Accumulator matrix and voting . . . 22
5.1.4 Find circle parameter with unknown radius . . . 22
5.2 Examples . . . 22
5.2.1 Find circles in shoeprint . . . 23
5.3 Limitations . . . 24
5.4 Extensions . . . 24
5.4.1 Adaptive Hough Transform . . . 24
5.5 Application . . . 24
5.5.1 People Counting . . . 24
5.6 Implementation code . . . 24
5.7 See also . . . 24
5.8 References . . . 24

6 RANSAC **25**

6.1 Example . . . 25
6.2 Overview . . . 25
6.3 Algorithm . . . 26
6.4 Matlab Implementation . . . 26
6.5 Parameters . . . 26
6.6 Advantages and disadvantages . . . 27
6.7 Applications . . . 27
6.8 Development and improvements . . . 27
6.9 Notes . . . 28
6.10 References . . . 28
6.11 External links . . . 29
6.12 Text and image sources, contributors, and licenses . . . 30
6.12.1 Text . . . 30
6.12.2 Images . . . 30
6.12.3 Content license . . . 31

Chapter 1

Edge detection

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed *edges*. The same problem of finding discontinuities in 1D signals is known as step detection and the problem of finding signal discontinuities over time is known as change detection. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction.[1]

1.1 Motivations

Canny edge detection applied to a photograph

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. It can be shown that under rather general assumptions for an image formation model, discontinuities in image brightness are likely to correspond to:[2][3]

- discontinuities in depth,
- discontinuities in surface orientation,
- changes in material properties and
- variations in scene illumination.

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. Thus, applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. However, it is not always possible to obtain such ideal edges from real life images of moderate complexity.

Edges extracted from non-trivial images are often hampered by *fragmentation*, meaning that the edge curves are not connected, missing edge segments as well as *false edges* not corresponding to interesting phenomena in the image – thus complicating the subsequent task of interpreting the image data.[4]

Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques.

1.2 Edge properties

The edges extracted from a two-dimensional image of a three-dimensional scene can be classified as either viewpoint dependent or viewpoint independent. A *viewpoint independent edge* typically reflects inherent properties of the three-dimensional objects, such as surface markings and surface shape. A *viewpoint dependent edge* may change as the viewpoint changes, and typically reflects the geometry of the scene, such as objects occluding one another.

A typical edge might for instance be the border between a block of red color and a block of yellow. In contrast a **line** (as can be extracted by a ridge detector) can be a small number of pixels of a different color on an otherwise unchanging background. For a line, there may therefore usually be one edge on each side of the line.

1.3 A simple edge model

Although certain literature has considered the detection of ideal step edges, the edges obtained from natural images are usually not at all ideal step edges. Instead they

are normally affected by one or several of the following effects:

- focal blur caused by a finite depth-of-field and finite point spread function.
- penumbral blur caused by shadows created by light sources of non-zero radius.
- shading at a smooth object

A number of researchers have used a Gaussian smoothed step edge (an error function) as the simplest extension of the ideal step edge model for modeling the effects of edge blur in practical applications.[4][5] Thus, a one-dimensional image f which has exactly one edge placed at $x = 0$ may be modeled as:

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l.$$

At the left side of the edge, the intensity is $I_l = \lim_{x\to-\infty} f(x)$, and right of the edge it is $I_r = \lim_{x\to\infty} f(x)$. The scale parameter σ is called the blur scale of the edge. Ideally this scale parameter should be adjusted based on the quality of image to avoid destroying true edges of the image.

1.4 Why edge detection is a non-trivial task

To illustrate why edge detection is not a trivial task, consider the problem of detecting edges in the following one-dimensional signal. Here, we may intuitively say that there should be an edge between the 4th and 5th pixels.

If the intensity difference were smaller between the 4th and the 5th pixels and if the intensity differences between the adjacent neighboring pixels were higher, it would not be as easy to say that there should be an edge in the corresponding region. Moreover, one could argue that this case is one in which there are several edges.

Hence, to firmly state a specific threshold on how large the intensity change between two neighbouring pixels must be for us to say that there should be an edge between these pixels is not always simple.[4] Indeed, this is one of the reasons why edge detection may be a non-trivial problem unless the objects in the scene are particularly simple and the illumination conditions can be well controlled (see for example, the edges extracted from the image with the girl above).

1.5 Approaches

There are many methods for edge detection, but most of them can be grouped into two categories, search-based and zero-crossing based. The search-based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied (see also noise reduction).

The edge detection methods that have been published mainly differ in the types of smoothing filters that are applied and the way the measures of edge strength are computed. As many edge detection methods rely on the computation of image gradients, they also differ in the types of filters used for computing gradient estimates in the x- and y-directions.

A survey of a number of different edge detection methods can be found in (Ziou and Tabbone 1998);[6] see also the encyclopedia articles on edge detection in *Encyclopedia of Mathematics*[3] and Encyclopedia of Computer Science and Engineering.[7]

1.5.1 Canny edge detection

Main article: Canny edge detector

John Canny considered the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge.[8] He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaussians. Canny also introduced the notion of non-maximum suppression, which means that given the presmoothing filters, edge points are defined as points where the gradient magnitude assumes a local maximum in the gradient direction. Looking for the zero crossing of the 2nd derivative along the gradient direction was first proposed by Haralick.[9] It took less than two decades to find a modern geometric variational meaning for that operator that links it to the Marr–Hildreth (zero crossing of the Laplacian) edge detector. That observation was presented by Ron Kimmel and Alfred Bruckstein.[10]

Although his work was done in the early days of computer vision, the Canny edge detector (including its variations) is still a state-of-the-art edge detector.[11] Unless the preconditions are particularly suitable, it is hard to find an edge detector that performs significantly better than the Canny edge detector.

The Canny-Deriche detector was derived from similar mathematical criteria as the Canny edge detector, although starting from a discrete viewpoint and then leading to a set of recursive filters for image smoothing instead of exponential filters or Gaussian filters.[12]

The differential edge detector described below can be seen as a reformulation of Canny's method from the viewpoint of differential invariants computed from a scale space representation leading to a number of advantages in terms of both theoretical analysis and sub-pixel implementation. In that aspect, Log Gabor filter have been shown to be a good choice to extract boundaries in natural scenes.[13]

1.5.2 Other first-order methods

Different gradient operators can be applied to estimate image gradients from the input image or a smoothed version of it. The simplest approach is to use central differences:

$$L_x(x,y) = -1/2 \cdot L(x-1,y) + 0 \cdot L(x,y) + 1/2 \cdot L(x+1,y)$$
$$L_y(x,y) = -1/2 \cdot L(x,y-1) + 0 \cdot L(x,y) + 1/2 \cdot L(x,y+1)$$

corresponding to the application of the following filter masks to the image data:

$$L_x = \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} * L.$$

The well-known and earlier Sobel operator is based on the following filters:

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * L.$$

Given such estimates of first-order image derivatives, the gradient magnitude is then computed as:

$$|\nabla L| = \sqrt{L_x^2 + L_y^2}$$

while the gradient orientation can be estimated as

$$\theta = \operatorname{atan2}(L_y, L_x).$$

Other first-order difference operators for estimating image gradient have been proposed in the Prewitt operator, Roberts cross and Frei-Chen.

It is possible to extend filters dimension to avoid the issue of recognizing edge in low SNR image. The cost of this operation is loss in terms of resolution. Examples are Extended Prewitt 7x7 and Abdou

1.5.3 Thresholding and linking

Once we have computed a measure of edge strength (typically the gradient magnitude), the next stage is to apply a threshold, to decide whether edges are present or not at an image point. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges.

If the edge thresholding is applied to just the gradient magnitude image, the resulting edges will in general be thick and some type of edge thinning post-processing is necessary. For edges detected with non-maximum suppression however, the edge curves are thin by definition and the edge pixels can be linked into edge polygon by an edge linking (edge tracking) procedure. On a discrete grid, the non-maximum suppression stage can be implemented by estimating the gradient direction using first-order derivatives, then rounding off the gradient direction to multiples of 45 degrees, and finally comparing the values of the gradient magnitude in the estimated gradient direction.

A commonly used approach to handle the problem of appropriate thresholds for thresholding is by using thresholding with hysteresis. This method uses multiple thresholds to find edges. We begin by using the upper threshold to find the start of an edge. Once we have a start point, we then trace the path of the edge through the image pixel by pixel, marking an edge whenever we are above the lower threshold. We stop marking our edge only when the value falls below our lower threshold. This approach makes the assumption that edges are likely to be in continuous curves, and allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge. Still, however, we have the problem of choosing appropriate thresholding parameters, and suitable thresholding values may vary over the image.

1.5.4 Edge thinning

Edge thinning is a technique used to remove the unwanted spurious points on the edges in an image. This technique is employed after the image has been filtered for noise (using median, Gaussian filter etc.), the edge operator has been applied (like the ones described above) to detect the edges and after the edges have been smoothed using an appropriate threshold value. This removes all the unwanted points and if applied carefully, results in one pixel thick edge elements.

Advantages:

1. Sharp and thin edges lead to greater efficiency in object recognition.

2. If Hough transforms are used to detect lines and ellipses, then thinning could give much better results.
3. If the edge happens to be the boundary of a region, then thinning could easily give the image parameters like perimeter without much algebra.

There are many popular algorithms used to do this, one such is described below:

1. Choose a type of connectivity, like 8, 6 or 4.
2. 8 connectivity is preferred, where all the immediate pixels surrounding a particular pixel are considered.
3. Remove points from North, south, east and west.
4. Do this in multiple passes, i.e. after the north pass, use the same semi processed image in the other passes and so on.
5. Remove a point if:
   The point has no neighbors in the North (if you are in the north pass, and respective directions for other passes).
   The point is not the end of a line.
   The point is isolated.
   Removing the points will not cause to disconnect its neighbors in any way.
6. Else keep the point.

The number of passes across direction should be chosen according to the level of accuracy desired.

1.5.5 Second-order approaches to edge detection

Some edge-detection operators are instead based upon second-order derivatives of the intensity. This essentially captures the rate of change in the intensity gradient. Thus, in the ideal continuous case, detection of zero-crossings in the second derivative captures local maxima in the gradient.

The early Marr-Hildreth operator is based on the detection of zero-crossings of the Laplacian operator applied to a Gaussian-smoothed image. It can be shown, however, that this operator will also return false edges corresponding to local minima of the gradient magnitude. Moreover, this operator will give poor localization at curved edges. Hence, this operator is today mainly of historical interest.

Differential edge detection

A more refined second-order edge detection approach which automatically detects edges with sub-pixel accuracy, uses the following *differential approach* of detecting zero-crossings of the second-order directional derivative in the gradient direction:

Following the differential geometric way of expressing the requirement of non-maximum suppression proposed by Lindeberg,[4][14] let us introduce at every image point a local coordinate system (u, v) , with the v -direction parallel to the gradient direction. Assuming that the image has been pre-smoothed by Gaussian smoothing and a scale space representation $L(x, y; t)$ at scale t has been computed, we can require that the gradient magnitude of the scale space representation, which is equal to the first-order directional derivative in the v -direction L_v , should have its first order directional derivative in the v -direction equal to zero

$$\partial_v(L_v) = 0$$

while the second-order directional derivative in the v -direction of L_v should be negative, i.e.,

$$\partial_{vv}(L_v) \leq 0.$$

Written out as an explicit expression in terms of local partial derivatives L_x , L_y ... L_{yyy} , this edge definition can be expressed as the zero-crossing curves of the differential invariant

$$L_v^2 L_{vv} = L_x^2\, L_{xx} + 2\, L_x\, L_y\, L_{xy} + L_y^2\, L_{yy} = 0,$$

that satisfy a sign-condition on the following differential invariant

$$L_v^3 L_{vvv} = L_x^3\, L_{xxx} + 3\, L_x^2\, L_y\, L_{xxy} + 3\, L_x\, L_y^2\, L_{xyy} + L_y^3\, L_{yyy} \leq 0$$

where L_x , L_y ... L_{yyy} denote partial derivatives computed from a scale space representation L obtained by smoothing the original image with a Gaussian kernel. In this way, the edges will be automatically obtained as continuous curves with sub-pixel accuracy. Hysteresis thresholding can also be applied to these differential and subpixel edge segments.

In practice, first-order derivative approximations can be computed by central differences as described above, while second-order derivatives can be computed from the scale space representation L according to:

$$L_{xx}(x, y) = L(x-1, y) - 2L(x, y) + L(x+1, y).$$

$$L_{xy}(x, y) = (L(x-1, y-1) - L(x-1, y+1) - L(x+1, y-1) + L(x+1, y+$$

$$L_{yy}(x, y) = L(x, y-1) - 2L(x, y) + L(x, y+1).$$

corresponding to the following filter masks:

$$L_{xx} = \begin{bmatrix}1 & -2 & 1\end{bmatrix} * L \quad \text{and} \quad L_{xy} = \begin{bmatrix}-1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4\end{bmatrix} * L \quad \text{and} \quad L_{yy} = \begin{bmatrix}1 \\ -2 \\ 1\end{bmatrix} * L.$$

Higher-order derivatives for the third-order sign condition can be obtained in an analogous fashion.

1.5.6 Phase congruency-based edge detection

A recent development in edge detection techniques takes a frequency domain approach to finding edge locations. Phase congruency (also known as phase coherence) methods attempt to find locations in an image where all sinusoids in the frequency domain are in phase. These locations will generally correspond to the location of a perceived edge, regardless of whether the edge is represented by a large change in intensity in the spatial domain. A key benefit of this technique is that it responds strongly to Mach bands, and avoids false positives typically found around roof edges. A roof edge, is a discontinuity in the first order derivative of a grey-level profile.[15]

1.5.7 Physics-inspired edge detection

Feature enhancement in an image (St Paul's Cathedral, London) using Phase Stretch Transform (PST). Left panel shows the original image and the right panel shows the detected features using PST.

The Phase Stretch Transform or PST is a physics-inspired computational approach to signal and image processing. One of its utilities is for feature detection and classification [16] .[17] PST is a spin-off from research on the Time stretch dispersive Fourier transform. PST transforms the image by emulating propagation through a diffractive medium with engineered 3D dispersive property (refractive index). The operation relies on symmetry of the dispersion profile and can be understood in terms of dispersive eigenfunctions or stretch modes.[18] PST performs similar functionality as phase contrast microscopy but on digital images. PST is also applicable to digital images as well as temporal, time series, data.

1.6 See also

- Convolution#Applications
- Feature detection (computer vision) for other low-level feature detectors
- Image derivatives
- Gabor filter
- Image noise reduction
- Kirsch operator for edge detection in the compass directions
- Ridge detection for relations between edge detectors and ridge detectors
- Log Gabor filter
- Phase Stretch Transform

1.7 References

[1] Umbaugh, Scott E (2010). *Digital image processing and analysis : human and computer vision applications with CVIPtools* (2nd ed.). Boca Raton, FL: CRC Press. ISBN 9-7814-3980-2052.

[2] H.G. Barrow and J.M. Tenenbaum (1981) "Interpreting line drawings as three-dimensional surfaces", Artificial Intelligence, vol 17, issues 1-3, pages 75-116.

[3] Lindeberg, Tony (2001), "Edge detection", in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

[4] T. Lindeberg (1998) "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pages 117-−154.

[5] W. Zhang and F. Bergholm (1997) "Multi-scale blur estimation and edge type classification for scene analysis", International Journal of Computer Vision, vol 24, issue 3, Pages: 219 - 250.

[6] D. Ziou and S. Tabbone (1998) "Edge detection techniques: An overview", International Journal of Pattern Recognition and Image Analysis, 8(4):537–559, 1998

[7] J. M. Park and Y. Lu (2008) "Edge detection in grayscale, color, and range images", in B. W. Wah (editor) Encyclopedia of Computer Science and Engineering, doi 10.1002/9780470050118.ecse603

[8] J. Canny (1986) "A computational approach to edge detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol 8, pages 679-714.

[9] R. Haralick, (1984) "Digital step edges from zero crossing of second directional derivatives", IEEE Trans. on Pattern Analysis and Machine Intelligence, 6(1):58–68.

[10] R. Kimmel and A.M. Bruckstein (2003) "On regularized Laplacian zero crossings and other optimal edge integrators", *International Journal of Computer Vision*, 53(3) pages 225-243.

[11] Shapiro L.G. & Stockman G.C. (2001) Computer Vision. London etc.: Prentice Hall, Page 326.

[12] R. Deriche (1987) *Using Canny's criteria to derive an optimal edge detector recursively implemented*, Int. J. Computer Vision, vol 1, pages 167–187.

[13] Sylvain Fischer, Rafael Redondo, Laurent Perrinet, Gabriel Cristobal. Sparse approximation of images inspired from the functional architecture of the primary visual areas. EURASIP Journal on Advances in Signal Processing, special issue on Image Perception, 2007

[14] T. Lindeberg (1993) "Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction", J. of Mathematical Imaging and Vision, 3(4), pages 349-−376.

[15] T. Pajdla and V. Hlavac (1993) "Surface discontinuities in range images," in Proc IEEE 4th Int. Conf. Comput. Vision, pp. 524-528.

[16] M. H. Asghari, and B. Jalali, "Edge detection in digital images using dispersive phase stretch," International Journal of Biomedical Imaging, Vol. 2015, Article ID 687819, pp. 1-6 (2015).

[17] M. H. Asghari, and B. Jalali, "Physics-inspired image edge detection," IEEE Global Signal and Information Processing Symposium (GlobalSIP 2014), paper: WdBD-L.1, Atlanta, December 2014.

[18] B. Jalali and A. Mahjoubfar, "Tailoring Wideband Signals With a Photonic Hardware Accelerator," Proceedings of the IEEE, Vol. 103, No. 7, pp. 1071-1086 (2015).

1.8 Further reading

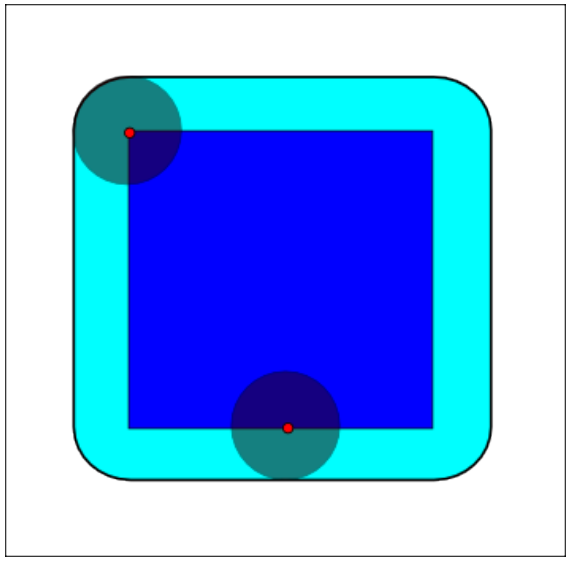
- Lindeberg, Tony (2001), "Edge detection", in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Entry on edge detection in Encyclopedia of Computer Science and Engineering
- Edge Detection using FPGA
- A-contrario line segment detection with code and on-line demonstration
- Edge detection using MATLAB

Chapter 2

Dilation (morphology)

Dilation is one of the basic operations in mathematical morphology. Originally developed for binary images, it has been expanded first to grayscale images, and then to complete lattices. The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image.

2.1 Binary operator



The dilation of dark-blue square by a disk, resulting in the light-blue square with rounded corners.

In binary morphology, dilation is a shift-invariant (translation invariant) operator, strongly related to the Minkowski addition.

A binary image is viewed in mathematical morphology as a subset of a Euclidean space $\mathbf{R}^d$ or the integer grid $\mathbf{Z}^d$, for some dimension d. Let E be a Euclidean space or an integer grid, A a binary image in E, and B a structuring element regarded as a subset of $\mathbf{R}^d$.

The dilation of A by B is defined by:

$$A \oplus B = \bigcup_{b \in B} A_b,$$

where Ab is the translation of A by b.

Dilation is commutative, also given by: $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$.

If B has a center on the origin, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. The dilation of a square of side 10, centered at the origin, by a disk of radius 2, also centered at the origin, is a square of side 14, with rounded corners, centered at the origin. The radius of the rounded corners is 2.

The dilation can also be obtained by: $A \oplus B = \{z \in E \mid (B^s)_z \cap A \neq \varnothing\}$, where B^s denotes the symmetric of B, that is, $B^s = \{x \in E \mid -x \in B\}$.

2.1.1 Properties of binary dilation

Here are some properties of the binary dilation operator

- It is translation invariant.
- It is increasing, that is, if $A \subseteq C$, then $A \oplus B \subseteq C \oplus B$.
- It is commutative.
- If the origin of E belongs to the structuring element B, then it is extensive, i.e., $A \subseteq A \oplus B$.
- It is associative, i.e., $(A \oplus B) \oplus C = A \oplus (B \oplus C)$.
- It is distributive over set union

2.2 Grayscale dilation

In grayscale morphology, images are functions mapping a Euclidean space or grid E into $\mathbb{R} \cup \{\infty, -\infty\}$, where $\mathbb{R}$ is the set of reals, ∞ is an element greater than any real number, and $-\infty$ is an element less than any real number.

Grayscale structuring elements are also functions of the same format, called "structuring functions".

Denoting an image by $f(x)$ and the structuring function by $b(x)$, the grayscale dilation of f by b is given by

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)],$$

where "sup" denotes the supremum.

2.2.1 Flat structuring functions

It is common to use flat structuring elements in morphological applications. Flat structuring functions are functions $b(x)$ in the form

$$b(x) = \begin{cases} 0, & x \in B, \\ -\infty, & \text{otherwise}, \end{cases}$$

where $B \subseteq E$.

In this case, the dilation is greatly simplified, and given by

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)] = \sup_{z \in E}[f(x - z) + b(z)] = \sup_{z \in B}[f(x - z)].$$

(Suppose $x = (px, qx)$, $z = (pz, qz)$, then $x - z = (px - pz, qx - qz)$.)

In the bounded, discrete case (E is a grid and B is bounded), the supremum operator can be replaced by the maximum. Thus, dilation is a particular case of order statistics filters, returning the maximum value within a moving window (the symmetric of the structuring function support B).

2.3 Dilation on complete lattices

Complete lattices are partially ordered sets, where every subset has an infimum and a supremum. In particular, it contains a least element and a greatest element (also denoted "universe").

Let $(L, \leq)$ be a complete lattice, with infimum and supremum symbolized by $\wedge$ and $\vee$, respectively. Its universe and least element are symbolized by U and $\varnothing$, respectively. Moreover, let $\{X_i\}$ be a collection of elements from L.

A dilation is any operator $\delta : L \to L$ that distributes over the supremum, and preserves the least element. I.e.:

- $\bigvee_i \delta(X_i) = \delta\left(\bigvee_i X_i\right)$,
- $\delta(\varnothing) = \varnothing$.

2.4 See also

- Buffer (GIS)
- Closing (morphology)
- Erosion (morphology)
- Mathematical morphology
- Opening (morphology)

2.5 Bibliography

- *Image Analysis and Mathematical Morphology* by Jean Serra, ISBN 0-12-637240-3 (1982)
- *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances* by Jean Serra, ISBN 0-12-637241-1 (1988)
- *An Introduction to Morphological Image Processing* by Edward R. Dougherty, ISBN 0-8194-0845-X (1992)

Chapter 3

Blob detection

In computer vision, **blob detection** methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other.

Given some property of interest expressed as a function of position on the image, there are two main classes of blob detectors: (i) *differential methods*, which are based on derivatives of the function with respect to position, and (ii) *methods based on local extrema*, which are based on finding the local maxima and minima of the function. With the more recent terminology used in the field, these detectors can also be referred to as *interest point operators*, or alternatively interest region operators (see also interest point detection and corner detection).

There are several motivations for studying and developing blob detectors. One main reason is to provide complementary information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of interest for further processing. These regions could signal the presence of objects or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is as main primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline stereo matching and to signal the presence of informative image features for appearance-based object recognition based on local image statistics. There is also the related notion of ridge detection to signal the presence of elongated objects.

3.1 The Laplacian of Gaussian

One of the first and also most common blob detectors is based on the Laplacian of the Gaussian (LoG). Given an input image $f(x, y)$, this image is convolved by a Gaussian kernel

$$g(x, y, t) = \frac{1}{2\pi t^2} e^{-\frac{x^2+y^2}{2t^2}}$$

at a certain scale t to give a scale space representation $L(x, y; t) = g(x, y, t) * f(x, y)$. Then, the result of applying the Laplacian operator

$$\nabla^2 L = L_{xx} + L_{yy}$$

is computed, which usually results in strong positive responses for dark blobs of extent $\sqrt{2t}$ and strong negative responses for bright blobs of similar size. A main problem when applying this operator at a single scale, however, is that the operator response is strongly dependent on the relationship between the size of the blob structures in the image domain and the size of the Gaussian kernel used for pre-smoothing. In order to automatically capture blobs of different (unknown) size in the image domain, a multi-scale approach is therefore necessary.

A straightforward way to obtain a *multi-scale blob detector with automatic scale selection* is to consider the *scale-normalized Laplacian operator*

$$\nabla^2_{norm} L(x, y; t) = t(L_{xx} + L_{yy})$$

and to detect *scale-space maxima/minima*, that are points that are *simultaneously local maxima/minima of* $\nabla^2_{norm} L$ *with respect to both space and scale* (Lindeberg 1994, 1998). Thus, given a discrete two-dimensional input image $f(x, y)$ a three-dimensional discrete scale-space volume $L(x, y, t)$ is computed and a point is regarded as a bright (dark) blob if the value at this point is greater (smaller) than the value in all its 26 neighbours. Thus, simultaneous selection of interest points $(\hat{x}, \hat{y})$ and scales $\hat{t}$ is performed according to

$$(\hat{x}, \hat{y}; \hat{t}) = \text{argmaxminlocal}_{(x,y;t)}(\nabla^2_{norm} L(x, y; t))$$

Note that this notion of blob provides a concise and mathematically precise operational definition of the notion of "blob", which directly leads to an efficient and robust

algorithm for blob detection. Some basic properties of blobs defined from scale-space maxima of the normalized Laplacian operator are that the responses are covariant with translations, rotations and rescalings in the image domain. Thus, if a scale-space maximum is assumed at a point $(x_0, y_0; t_0)$ then under a rescaling of the image by a scale factor s , there will be a scale-space maximum at $(sx_0, sy_0; s^2t_0)$ in the rescaled image (Lindeberg 1998). This in practice highly useful property implies that besides the specific topic of Laplacian blob detection, *local maxima/minima of the scale-normalized Laplacian are also used for scale selection in other contexts*, such as in corner detection, scale-adaptive feature tracking (Bretzner and Lindeberg 1998), in the scale-invariant feature transform (Lowe 2004) as well as other image descriptors for image matching and object recognition.

The scale selection properties of the Laplacian operator and other closely scale-space interest point detectors are analyzed in detail in (Lindeberg 2013a).[1] In (Lindeberg 2013b, 2015)[2][3] it is shown that there exist other scale-space interest point detectors, such as the determinant of the Hessian operator, that perform better than Laplacian operator or its difference-of-Gaussians approximation for image-based matching using local SIFT-like image descriptors.

3.2 The difference of Gaussians approach

From the fact that the scale space representation $L(x, y, t)$ satisfies the diffusion equation

$$\partial_t L = \frac{1}{2}\nabla^2 L$$

it follows that the Laplacian of the Gaussian operator $\nabla^2 L(x, y, t)$ can also be computed as the limit case of the difference between two Gaussian smoothed images (scale space representations)

$$\nabla^2_{norm} L(x, y; t) \approx \frac{t}{\Delta t}\left(L(x, y; t + \Delta t) - L(x, y; t)\right)$$

In the computer vision literature, this approach is referred to as the difference of Gaussians (DoG) approach. Besides minor technicalities, however, this operator is in essence similar to the Laplacian and can be seen as an approximation of the Laplacian operator. In a similar fashion as for the Laplacian blob detector, blobs can be detected from scale-space extrema of differences of Gaussians—see (Lindeberg 2012, 2015)[4][3] for the explicit relation between the difference-of-Gaussian operator and the scale-normalized Laplacian operator. This approach is for instance used in the scale-invariant feature transform (SIFT) algorithm—see Lowe (2004).

3.3 The determinant of the Hessian

By considering the scale-normalized determinant of the Hessian, also referred to as the Monge–Ampère operator,

$$\det HL(x, y; t) = t^2(L_{xx}L_{yy} - L^2_{xy})$$

where HL denotes the Hessian matrix of L and then detecting scale-space maxima of this operator one obtains another straightforward differential blob detector with automatic scale selection which also responds to saddles (Lindeberg 1994, 1998)

$$(\hat{x}, \hat{y}; \hat{t}) = \text{argmaxlocal}_{(x,y;t)}(\det HL(x, y; t))$$

The blob points $(\hat{x}, \hat{y})$ and scales $\hat{t}$ are also defined from an operational differential geometric definitions that leads to blob descriptors that are covariant with translations, rotations and rescalings in the image domain. In terms of scale selection, blobs defined from scale-space extrema of the determinant of the Hessian (DoH) also have slightly better scale selection properties under non-Euclidean affine transformations than the more commonly used Laplacian operator (Lindeberg 1994, 1998). In simplified form, the scale-normalized determinant of the Hessian computed from Haar wavelets is used as the basic interest point operator in the SURF descriptor (Bay et al. 2006) for image matching and object recognition.

A detailed analysis of the selection properties of the determinant of the Hessian operator and other closely scale-space interest point detectors is given in (Lindeberg 2013a)[1] showing that the determinant of the Hessian operator has better scale selection properties under affine image transformations than the Laplacian operator. In (Lindeberg 2013b, 2015)[2][3] it is shown that the determinant of the Hessian operator performs significantly better than the Laplacian operator or its difference-of-Gaussians approximation for image-based matching using local SIFT-like or SURF-like image descriptors, leading to higher efficiency values and lower 1-precision scores.

3.4 The hybrid Laplacian and determinant of the Hessian operator (Hessian-Laplace)

A hybrid operator between the Laplacian and the determinant of the Hessian blob detectors has also been proposed, where spatial selection is done by the determinant of the Hessian and scale selection is performed with the scale-normalized Laplacian (Mikolajczyk and Schmid 2004):

$$(\hat{x}, \hat{y}) = \text{argmaxlocal}_{(x,y)}(\det HL(x, y; t))$$

$$\hat{t} = \text{argmaxminlocal}_t(\nabla^2_{norm} L(\hat{x}, \hat{y}; t))$$

This operator has been used for image matching, object recognition as well as texture analysis.

3.5 Affine-adapted differential blob detectors

The blob descriptors obtained from these blob detectors with automatic scale selection are invariant to translations, rotations and uniform rescalings in the spatial domain. The images that constitute the input to a computer vision system are, however, also subject to perspective distortions. To obtain blob descriptors that are more robust to perspective transformations, a natural approach is to devise a blob detector that is *invariant to affine transformations*. In practice, affine invariant interest points can be obtained by applying affine shape adaptation to a blob descriptor, where the shape of the smoothing kernel is iteratively warped to match the local image structure around the blob, or equivalently a local image patch is iteratively warped while the shape of the smoothing kernel remains rotationally symmetric (Lindeberg and Garding 1997; Baumberg 2000; Mikolajczyk and Schmid 2004, Lindeberg 2008/2009). In this way, we can define affine-adapted versions of the Laplacian/Difference of Gaussian operator, the determinant of the Hessian and the Hessian-Laplace operator (see also Harris-Affine and Hessian-Affine).

3.6 Grey-level blobs, grey-level blob trees and scale-space blobs

A natural approach to detect blobs is to associate a bright (dark) blob with each local maximum (minimum) in the intensity landscape. A main problem with such an approach, however, is that local extrema are very sensitive to noise. To address this problem, Lindeberg (1993, 1994) studied the problem of detecting local maxima with extent at multiple scales in scale space. A region with spatial extent defined from a watershed analogy was associated with each local maximum, as well a local contrast defined from a so-called delimiting saddle point. A local extremum with extent defined in this way was referred to as a *grey-level blob*. Moreover, by proceeding with the watershed analogy beyond the delimiting saddle point, a *grey-level blob tree* was defined to capture the nested topological structure of level sets in the intensity landscape, in a way that is invariant to affine deformations in the image domain and monotone intensity transformations. By studying how these structures evolve with increasing scales, the notion of *scale-space blobs* was introduced. Beyond local contrast and extent, these scale-space blobs also measured how stable image structures are in scale-space, by measuring their *scale-space lifetime*.

It was proposed that regions of interest and scale descriptors obtained in this way, with associated scale levels defined from the scales at which normalized measures of blob strength assumed their maxima over scales could be used for guiding other early visual processing. An early prototype of simplified vision systems was developed where such regions of interest and scale descriptors were used for directing the focus-of-attention of an active vision system. While the specific technique that was used in these prototypes can be substantially improved with the current knowledge in computer vision, the overall general approach is still valid, for example in the way that local extrema over scales of the scale-normalized Laplacian operator are nowadays used for providing scale information to other visual processes.

3.6.1 Lindeberg's watershed-based grey-level blob detection algorithm

For the purpose of detecting *grey-level blobs* (local extrema with extent) from a watershed analogy, Lindeberg developed an algorithm based on *pre-sorting* the pixels, alternatively connected regions having the same intensity, in decreasing order of the intensity values. Then, comparisons were made between nearest neighbours of either pixels or connected regions.

For simplicity, let us consider the case of detecting bright grey-level blobs and let the notation "higher neighbour" stand for "neighbour pixel having a higher grey-level value". Then, at any stage in the algorithm (carried out in decreasing order of intensity values) is based on the following classification rules:

1. If a region has no higher neighbour, then it is a local maximum and will be the seed of a blob.

2. Else, if it has at least one higher neighbour, which is background, then it cannot be part of any blob and must be background.

3. Else, if it has more than one higher neighbour and if those higher neighbours are parts of different blobs, then it cannot be a part of any blob, and must be background.

4. Else, it has one or more higher neighbours, which are all parts of the same blob. Then, it must also be a part of that blob.

Compared to other watershed methods, the flooding in this algorithm stops once the intensity level falls below the intensity value of the so-called *delimiting saddle point* associated with the local maximum. However, it is rather

straightforward to extend this approach to other types of watershed constructions. For example, by proceeding beyond the first delimiting saddle point a "grey-level blob tree" can be constructed. Moreover, the grey-level blob detection method was embedded in a scale space representation and performed at all levels of scale, resulting in a representation called the *scale-space primal sketch*.

This algorithm with its applications in computer vision is described in more detail in Lindeberg's thesis [5] as well as the monograph on scale-space theory [6] partially based on that work. Earlier presentations of this algorithm can also be found in.[7][8] More detailed treatments of applications of grey-level blob detection and the scale-space primal sketch to computer vision and medical image analysis are given in.[9][10][11]

3.7 Maximally stable extremum regions (MSER)

Main article: Maximally stable extremal regions

Matas et al. (2002) were interested in defining image descriptors that are robust under perspective transformations. They studied level sets in the intensity landscape and measured how stable these were along the intensity dimension. Based on this idea, they defined a notion of *maximally stable extremum regions* and showed how these image descriptors can be used as image features for stereo matching.

There are close relations between this notion and the above-mentioned notion of grey-level blob tree. The maximally stable extremum regions can be seen as making a specific subset of the grey-level blob tree explicit for further processing.

3.8 See also

- Blob extraction
- Corner detection
- Affine shape adaptation
- Scale space
- Ridge detection
- Interest point detection
- Feature detection (computer vision)
- Harris-Affine
- Hessian-Affine
- PCBR

3.9 References

- Christopher Evans. "Notes on the OpenSURF library".
- H. Bay, T. Tuytelaars and L. van Gool (2006). "SURF: Speeded Up Robust Features". *Proceedings of the 9th European Conference on Computer Vision, Springer LNCS volume 3951, part 1*. pp. 404–417.
- L. Bretzner and T. Lindeberg (1998). "Feature Tracking with Automatic Selection of Spatial Scales" (abstract page). *Computer Vision and Image Understanding* **71** (3): 385–392. doi:10.1006/cviu.1998.0650.
- T. Lindeberg (1993). "Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-of-Attention" (abstract page). *International Journal of Computer Vision* **11** (3): 283–318. doi:10.1007/BF01469346.
- T. Lindeberg (1994). *Scale-Space Theory in Computer Vision*. Springer. ISBN 0-7923-9418-6.
- T. Lindeberg (1998). "Feature detection with automatic scale selection" (abstract page). *International Journal of Computer Vision* **30** (2): 77–116. doi:10.1023/A:1008045108935.
- Lindeberg, T.; Garding, J. (1997). "Shape-adapted smoothing in estimation of 3-{D} depth cues from affine distortions of local 2-{D} structure". *Image and Vision Computing* **15** (6): 415–434. doi:10.1016/S0262-8856(97)01144-X.
- Lindeberg, T. (2008). "Scale-space". In Wah, Benjamin. *Encyclopedia of Computer Science and Engineering* **IV**. John Wiley and Sons. pp. 2495–2504. doi:10.1002/9780470050118.ecse609. ISBN 0-470-05011-X.
- D. G. Lowe (2004). "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision* **60** (2): 91–110. doi:10.1023/B:VISI.0000029664.99615.94.
- J. Matas, O. Chum, M. Urban and T. Pajdla (2002). "Robust wide baseline stereo from maximally stable extremum regions" (PDF). *British Machine Vision Conference*. pp. 384–393.
- K. Mikolajczyk, K. and C. Schmid (2004). "Scale and affine invariant interest point detectors" (PDF). *International Journal of Computer Vision* **60** (1): 63–86. doi:10.1023/B:VISI.0000027790.02288.f2.

[1] Lindeberg, Tony (2013) "Scale Selection Properties of Generalized Scale-Space Interest Point Detectors", Journal of Mathematical Imaging and Vision, Volume 46, Issue 2, pages 177-210.

[2] Lindeberg (2013) “Image Matching Using Generalized Scale-Space Interest Points”, Scale Space and Variational Methods in Computer Vision, Springer Lecture Notes in Computer Science Volume 7893, 2013, pp 355-367.

[3] T. Lindeberg ``Image matching using generalized scale-space interest points”, Journal of Mathematical Imaging and Vision, volume 52, number 1, pages 3-36, 2015.

[4] T. Lindeberg ``Scale invariant feature transform, *Scholarpedia, 7(5):10491, 2012.*

[5] Lindeberg, T. (1991) *Discrete Scale-Space Theory and the Scale-Space Primal Sketch*, PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden, May 1991. (ISSN 1101-2250. ISRN KTH NA/P-−91/8--SE) (The grey-level blob detection algorithm is described in section 7.1)

[6] Lindeberg, Tony, *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, 1994, ISBN 0-7923-9418-6

[7] T. Lindeberg and J.-O. Eklundh, “Scale detection and region extraction from a scale-space primal sketch”, in *Proc. 3rd International Conference on Computer Vision*, (Osaka, Japan), pp. 416-−426, Dec. 1990. (See Appendix A.1 for the basic definitions for the watershed-based grey-level blob detection algorithm.)

[8] T. Lindeberg and J.-O. Eklundh, “On the computation of a scale-space primal sketch”, *Journal of Visual Communication and Image Representation*, vol. 2, pp. 55-−78, Mar. 1991.

[9] Lindeberg, T.: Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention, *International Journal of Computer Vision*, 11(3), 283-−318, 1993.

[10] Lindeberg, T, Lidberg, Par and Roland, P. E..: “Analysis of Brain Activation Patterns Using a 3-D Scale-Space Primal Sketch”, *Human Brain Mapping*, vol 7, no 3, pp 166-−194, 1999.

[11] Jean-Francois Mangin, Denis Rivière, Olivier Coulon, Cyril Poupon, Arnaud Cachia, Yann Cointepas, Jean-Baptiste Poline, Denis Le Bihan, Jean Régis, Dimitri Papadopoulos-Orfanos: “Coordinate-based versus structural approaches to brain image analysis”. *Artificial Intelligence in Medicine* 30(2): 177-197 (2004)

Chapter 4

Hough transform

The **Hough transform** 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][4] 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".

4.1 History

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).

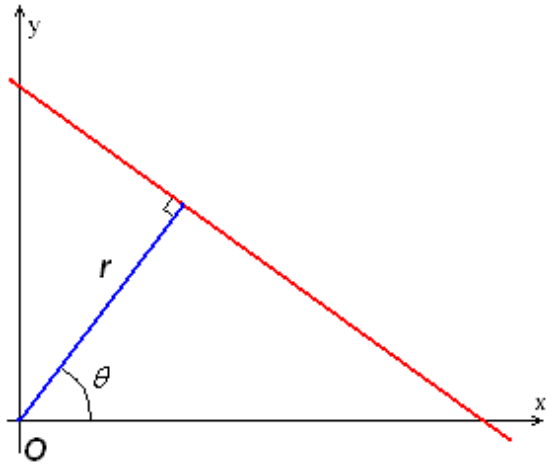
4.2 Theory

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 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[5] 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 θ (*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, θ) . The (r, θ) 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.[6])

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,θ) 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,θ) for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves.[7]

4.3 Implementation

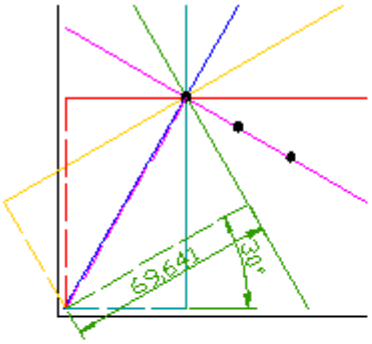
The linear Hough transform algorithm uses a two-dimensional array, called an accumulator, to detect the existence of a line described by $r = x \cos\theta + y \sin\theta$. The dimension of the accumulator equals the number of unknown parameters, i.e., two, considering quantized values of r and θ in the pair (r,θ). 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,θ) 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.

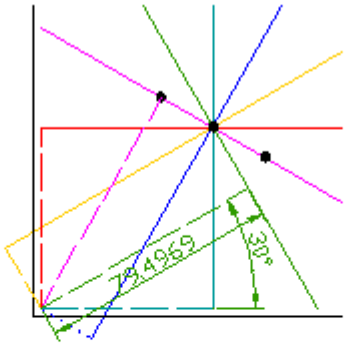
The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator—one dimension of this matrix is the quantized angle θ and the other dimension is the quantized distance r. Each element of the matrix has a value equal to the number of points or pixels that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

4.4 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

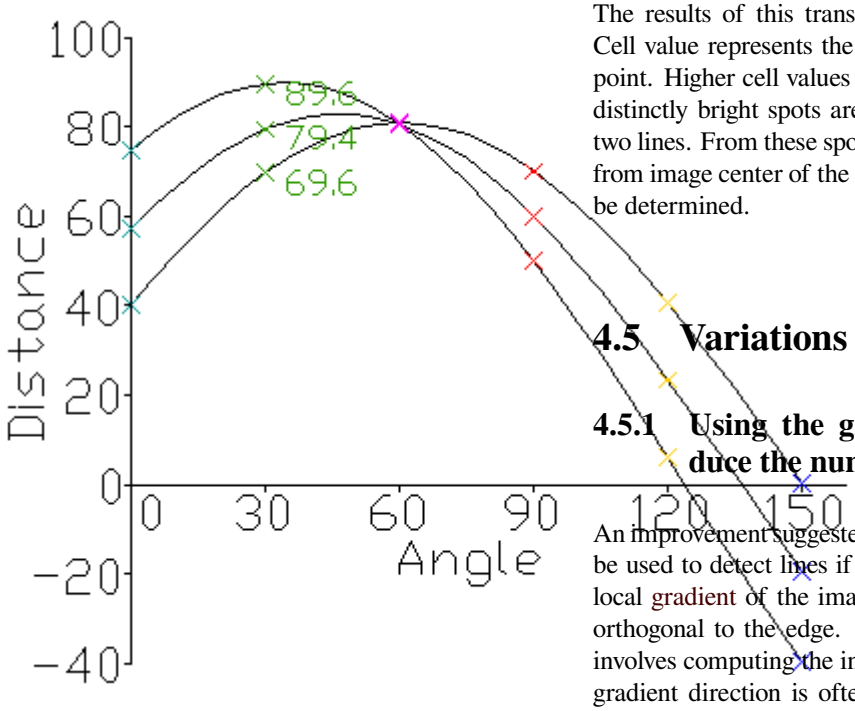


Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

- 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.

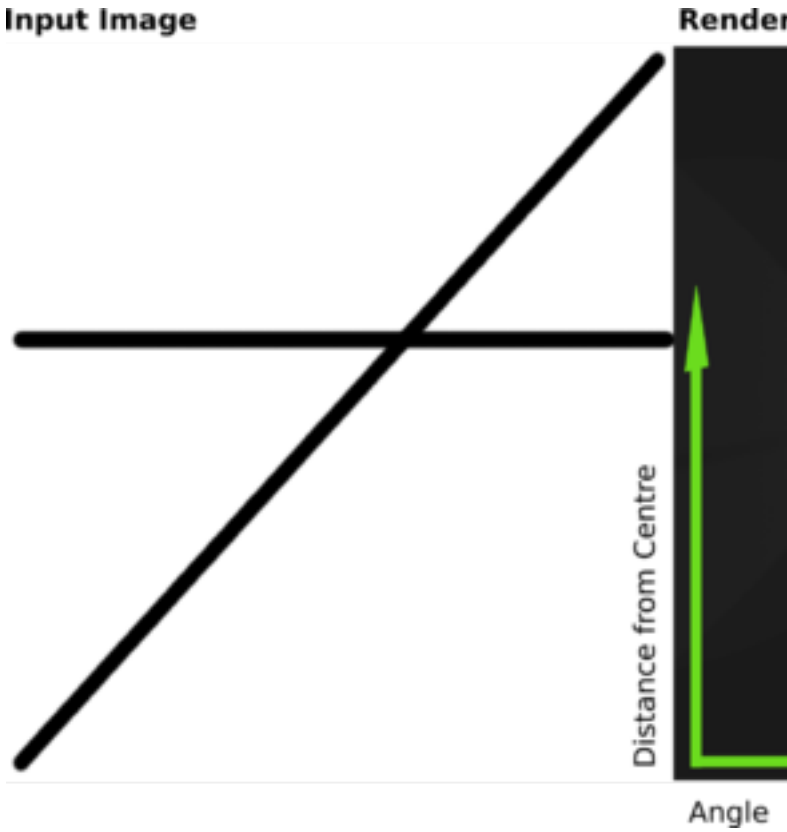
- This is repeated for each data point.

- 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.

4.5 Variations and extensions

4.5.1 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.

4.5.2 Kernel-based Hough transform (KHT)

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, 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.

4.5.3 3-D Kernel-based Hough transform for plane detection (3DKHT)

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 (θ, ϕ, ρ) 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.

4.5.4 Hough transform of curves, and its generalization for analytical and non-analytical shapes

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.

4.5.5 Circle detection process

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.

4.5.6 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 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 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.[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.[16]

4.5.7 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.

4.5.8 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.

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 give an efficient way of implementing the Hough transform for ellipse detection by overcoming the memory issues.[17] 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.

4.6 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.[18]

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 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.

4.7 See also

- Generalised Hough transform
- Randomized Hough transform
- Radon transform
- Fourier transform

4.8 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] Hough, P.V.C. *Method and means for recognizing complex patterns,* U.S. Patent 3,069,654, Dec. 18, 1962

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

[5] 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).

[6] CiteSeerX — A short introduction to the Radon and Hough transforms and how they relate to each other

[7] "Hough Transform".

[8] Jensen, Jeppe. "Hough Transform for Straight Lines" (PDF). Retrieved 16 December 2011.

[9] 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.

[10] 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.

[11] 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.

[12] Ballard, D.H. (1981). "Generalizing the Houghtransform to detectarbitraryshapes". *Pattern Recognition* **13** (2): 111–122. doi:10.1016/0031-3203(81)90009-1.

[13] 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.

[14] 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

[15] 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.

[16] 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.

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

[18] "Image Transforms - Hough Transform". Homepages.inf.ed.ac.uk. Retrieved 2009-08-17.

4.9 External links

- hough_transform.cpp – C++ code – example of CImg 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.
- 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.

Chapter 5

Circle Hough Transform

The **circle Hough Transform** (**CHT**) is a feature extraction technique for detecting circles. It is a specialization of Hough Transform. The purpose of the technique is to find circles in imperfect image inputs. The circle candidates are produced by "voting" in the Hough parameter space and then select the local maxima in a so-called accumulator matrix.

5.1 Theory

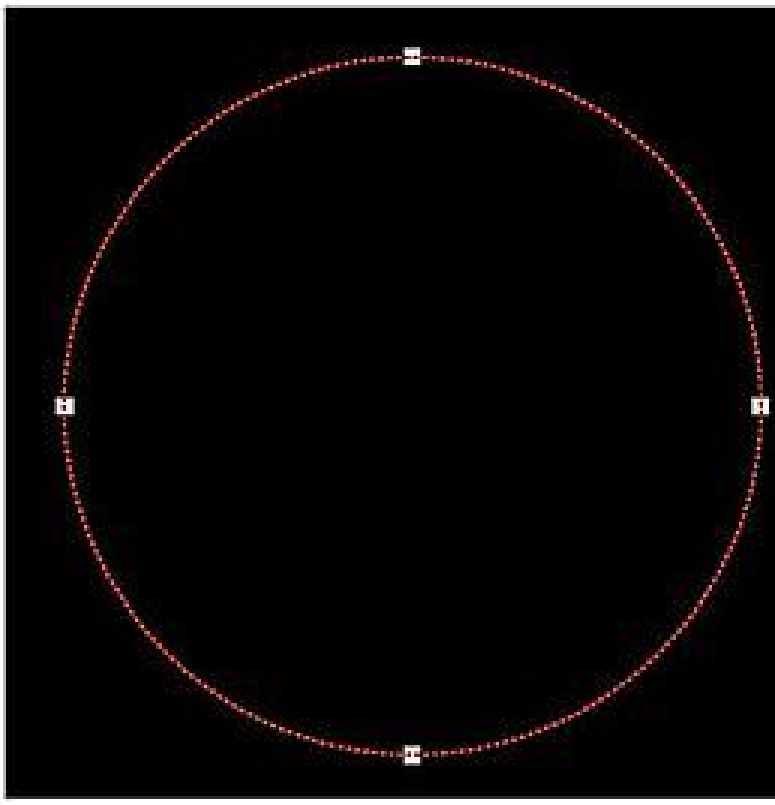
In a two dimensional space, a circle can by described by:

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

where (a,b) is the center of the circle, and r is the radius. If a 2D point (x,y) is fixed, then the parameters can be found according to (1). The parameter space would be three dimensional, (a, b, r). And all the parameters that satisfying (x, y) would lie on the surface of an inverted right-angled cone whose apex is at (x, y, 0). In the 3D space, the circle parameters can be identified by the intersection of many conic surfaces that are defined by points on the 2D circle. This process can be divided into two stages. The first stage is fixing radius then find the optimal center of circles in a 2D parameter space. The second stage is to find the optimal radius in a one dimensional parameter space.

5.1.1 Find parameters with Known radius R

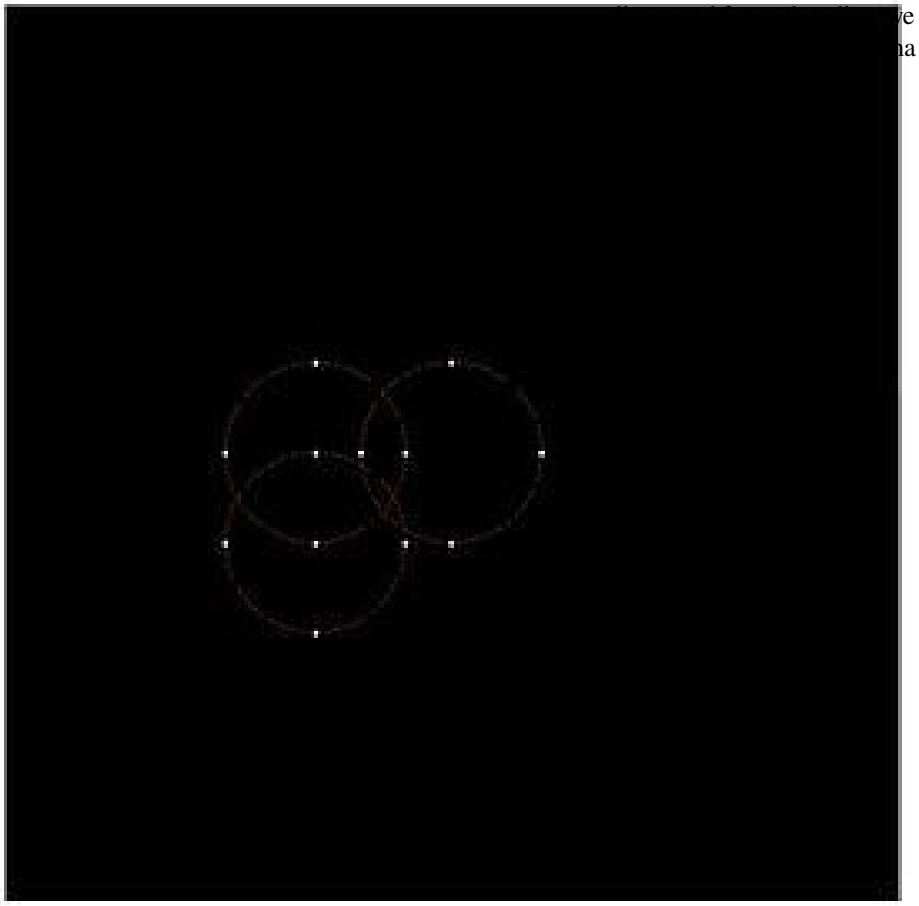
If the radius is fixed, then the parameter space would be reduced to 2D (the position of the circle center). For each point (x, y) on the original circle, it can define a circle centered at (x, y) with radius R according to (1). The intersection point of all such circles in the parameter space would be corresponding to the center point of the original circle.



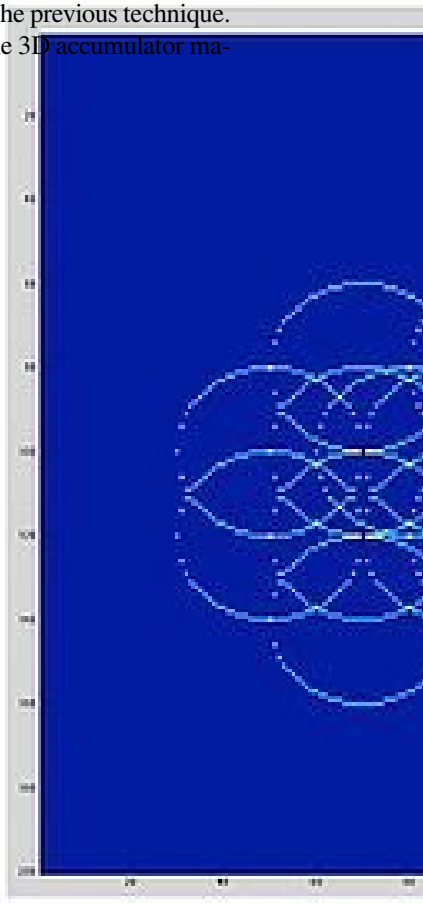
Consider 4 points on a circle in the original image (left). The circle Hough transform is shown in the right. Note that the radius is assumed to be known. For each (x,y) of the four points (white points) in the original image, it can define a circle in the Hough parameter space centered at (x, y) with radius r. An accumulator matrix is used for tracking the intersection point. In the parameter space, the voting number of points through which the circle passing would be increased by one. Then the local maxima point (the red point in the center in the right figure) can be found. The position (a, b) of the maxima would be the center of the original circle.

5.1.2 Multiple circles with known radius R

Multiple circles with same radius can be found with the same technique.



ve use the previous technique.
a in the 3D accumulator ma-



Note that, in the accumulator matrix (right fig), there would be at least 3 local maxima points.

5.1.3 Accumulator matrix and voting

In practice, an accumulator matrix is introduced to find the intersection point in the parameter space. First, we need to divide the parameter space into "buckets" using a grid and produce an accumulator matrix according to the grid. The element in the accumulator matrix denotes the number of "circles" in the parameter space that passing through the corresponding grid cell in the parameter space. The number is also called "voting number". Initially, every element in the matrix is zeros. Then for each "edge" point in the original space, we can formulate a circle in the parameter space and increase the voting number of the grid cell which the circle passing through. This process is called "voting".

After voting, we can find local maxima in the accumulator matrix. The positions of the local maxima are corresponding to the circle centers in the original space.

5.1.4 Find circle parameter with unknown radius

Since the parameter space is 3D, the accumulator matrix would be 3D, too. We can iterate through possible ra-

5.2 Examples

5.2.1 Find circles in shoeprint

The original picture (right) is first turned into a binary image (left) using a threshold and Gaussian filter. Then edges (mid) are found from it using canny edge detection. After this, all the edge points are used by the Circle Hough Transform to find underlying circle structure.

5.3 Limitations

Since the parameter space of CHT is three dimensional, it may require lots of storage and computation. Choosing bigger grid size can ameliorate this problem.

However, choose an appropriate grid size is difficult. Since too coarse a grid can lead to large values of the vote being obtained falsely because many quite different structures correspond to a single bucket. Too fine a grid can lead to structures not being found because votes resulting from tokens that are not exactly aligned end up in different buckets, and no bucket has a large vote.

Also, CHT is not very robust to noise.

5.4 Extensions

5.4.1 Adaptive Hough Transform

J. Illingworth and J. Kittler [1] introduced this method for implementing Hough Transform efficiently. The AHT uses a small accumulator array and the idea of a flexible iterative "coarse to fine" accumulation and search strategy to identify significant peaks in the Hough parameter spaces. This method is substantially superior to the standard Hough Transform implementation in both storage and computational requirements.

5.5 Application

5.5.1 People Counting

Since the head would be similar to a circle in an image. CHT can be used for detecting heads in a picture, so as to count the number of persons in the image.[2]

5.6 Implementation code

- http://www.mathworks.com/matlabcentral/fileexchange/4985-circle-detection-via-standard-hough-transform
- http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html

5.7 See also

- Hough transform
- Generalised Hough transform
- Randomized Hough transform
- https://www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf

5.8 References

[1] J. Illingworth and J. Kittler, "The Adaptive Hough Transform," PAMI-9 , Issue: 5, 1987, pp 690-698

[2] Hong Liu, Yueliang Qian and Shouxun Lin , "DETECTING PERSONS USING HOUGH CIRCLE TRANSFORM IN SURVEILLANCE VIDEO"

Chapter 6

RANSAC

Random sample consensus (**RANSAC**) is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. Therefore, it also can be interpreted as an outlier detection method.[1] It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. The algorithm was first published by Fischler and Bolles at SRI International in 1981.They used RANSAC to solve the Location Determination Problem (LDP), where the goal is to determine the points in the space that project onto an image into a set of landmarks with known locations.

A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outliers" which are data that do not fit the model. The outliers can come, e.g., from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.

6.1 Example

A simple example is fitting of a line in two dimensions to a set of observations. Assuming that this set contains both inliers, i.e., points which approximately can be fitted to a line, and outliers, points which cannot be fitted to this line, a simple least squares method for line fitting will generally produce a line with a bad fit to the inliers. The reason is that it is optimally fitted to all points, including the outliers. RANSAC, on the other hand, can produce a model which is only computed from the inliers, provided that the probability of choosing only inliers in the selection of data is sufficiently high. There is no guarantee for this situation, however, and there are a number of algorithm parameters which must be carefully chosen to keep the level of probability reasonably high.

- A data set with many outliers for which a line has to be fitted.
- Fitted line with RANSAC; outliers have no influence on the result.

6.2 Overview

The RANSAC algorithm is a learning technique to estimate parameters of a model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: that the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model (few missing data). The RANSAC algorithm is essentially composed of two steps that are iteratively repeated:

1. In the first step, a sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.
2. In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers.

The input to the RANSAC algorithm is a set of observed data values, a way of fitting some kind of model to the observations, and some confidence parameters. RANSAC achieves its goal by repeating the following steps:

1. Select a random subset of the original data. Call this subset the *hypothetical inliers*.
2. A model is fitted to the set of hypothetical inliers.
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the *consensus set*.
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.
5. Afterwards, the model may be improved by reestimating it using all members of the consensus set.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

- RANSAC: Inliers and Outliers.

6.3 Algorithm

The generic RANSAC algorithm works as follows:

Given: data - a set of observed data points model - a model that can be fitted to data points n - the minimum number of data values required to fit the model k - the maximum number of iterations allowed in the algorithm t - a threshold value for determining when a data point fits a model d - the number of close data values required to assert that a model fits well to data Return: bestfit - model parameters which best fit the data (or nil if no good model is found) iterations = 0 bestfit = nil besterr = something really large while iterations < k { maybeinliers = n randomly selected values from data maybemodel = model parameters fitted to maybeinliers alsoinliers = empty set for every point in data not in maybeinliers { if point fits maybemodel with an error smaller than t add point to alsoinliers } if the number of elements in alsoinliers is > d { % this implies that we may have found a good model % now test how good it is bettermodel = model parameters fitted to all points in maybeinliers and alsoinliers thiserr = a measure of how well model fits these points if thiserr < besterr { bestfit = bettermodel besterr = thiserr } } increment iterations } return bestfit

6.4 Matlab Implementation

The matlab implementation of 2D line fitting using RANSAC algorithm:

function [bestParameter1,bestParameter2] = ransac_demo(data,num,iter,threshDist,inlierRatio) % data: a 2xn dataset with #n data points % num: the minimum number of points. For line fitting problem, num=2 % iter: the number of iterations % threshDist: the threshold of the distances between points and the fitting line % inlierRatio: the threshold of the number of inliers %% Plot the data points figure;plot(data(1,:),data(2,:),'o');hold on; number = size(data,2); % Total number of points bestInNum = 0; % Best fitting line with largest number of inliers bestParameter1=0;bestParameter2=0; % parameters for best fitting line for i=1:iter %% Randomly select 2 points idx = randperm(number,num); sample = data(:,idx); %% Compute the distances between all points with the fitting line kLine = sample(:,2)-sample(:,1); kLineNorm = kLine/norm(kLine); normVector = [-kLineNorm(2),kLineNorm(1)]; distance = normVector*(data - repmat(sample(:,1),1,number)); %% Compute the inliers with distances smaller than the threshold inlierIdx = find(abs(distance)<=threshDist); inlierNum = length(inlierIdx); %% Update the number of inliers and fitting model if better model is found if inlierNum>=round(inlierRatio*number) && inlierNum>bestInNum bestInNum = inlierNum; parameter1 = (sample(2,2)-sample(2,1))/(sample(1,2)-sample(1,1)); parameter2 = sample(2,1)-parameter1*sample(1,1); bestParameter1=parameter1; bestParameter2=parameter2; end end %% Plot the best fitting line xAxis = -number/2:number/2; yAxis = bestParameter1*xAxis + bestParameter2; plot(xAxis,yAxis,'r-','LineWidth',2);

6.5 Parameters

The values of parameters t and d have to be determined from specific requirements related to the application and the data set, possibly based on experimental evaluation. The parameter k (the number of iterations), however, can be determined from a theoretical result. Let p be the probability that the RANSAC algorithm in some iteration selects only inliers from the input data set when it chooses the n points from which the model parameters are estimated. When this happens, the resulting model is likely to be useful so p gives the probability that the algorithm produces a useful result. Let w be the probability of choosing an inlier each time a single point is selected, that is,

w = number of inliers in data / number of points in data

A common case is that w is not well known beforehand, but some rough value can be given. Assuming that the n points needed for estimating a model are selected independently, w^n is the probability that all *n* points are in-

liers and $1-w^n$ is the probability that at least one of the n points is an outlier, a case which implies that a bad model will be estimated from this point set. That probability to the power of k is the probability that the algorithm never selects a set of n points which all are inliers and this must be the same as $1-p$. Consequently,

$$1-p=(1-w^n)^k$$

which, after taking the logarithm of both sides, leads to

$$k=\frac{\log(1-p)}{\log(1-w^n)}$$

This result assumes that the n data points are selected independently, that is, a point which has been selected once is replaced and can be selected again in the same iteration. This is often not a reasonable approach and the derived value for k should be taken as an upper limit in the case that the points are selected without replacement. For example, in the case of finding a line which fits the data set illustrated in the above figure, the RANSAC algorithm typically chooses two points in each iteration and computes maybe_model as the line between the points and it is then critical that the two points are distinct.

To gain additional confidence, the standard deviation or multiples thereof can be added to k. The standard deviation of k is defined as

$$SD(k)=\frac{\sqrt{1-w^n}}{w^n}$$

6.6 Advantages and disadvantages

An advantage of RANSAC is its ability to do robust estimation[2] of the model parameters, i.e., it can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. A disadvantage of RANSAC is that there is no upper bound on the time it takes to compute these parameters (except exhaustion). When the number of iterations computed is limited the solution obtained may not be optimal, and it may not even be one that fits the data in a good way. In this way RANSAC offers a trade-off; by computing a greater number of iterations the probability of a reasonable model being produced is increased. Moreover, RANSAC is not always able to find the optimal set even for moderately contaminated sets and it usually performs badly when the number of inliers is less than 50%. Optimal RANSAC [3] was proposed to handle both these problems and is capable of finding the optimal set for heavily contaminated sets, even for an inlier ratio under 5%. Another disadvantage of RANSAC is that it requires the setting of problem-specific thresholds.

RANSAC can only estimate one model for a particular data set. As for any one-model approach when two (or more) model instances exist, RANSAC may fail to find either one. The Hough transform is one alternative robust estimation technique that may be useful when more than one model instance is present. Another approach for multi model fitting is known as PEARL,[4] which combines model sampling from data points as in RANSAC with iterative re-estimation of inliers and the multi-model fitting being formulated as an optimization problem with a global energy functional describing the quality of the overall solution.

6.7 Applications

The RANSAC algorithm is often used in computer vision, e.g., to simultaneously solve the correspondence problem and estimate the fundamental matrix related to a pair of stereo cameras.

6.8 Development and improvements

Since 1981 RANSAC has become a fundamental tool in the computer vision and image processing community. In 2006, for the 25th anniversary of the algorithm, a workshop was organized at the International Conference on Computer Vision and Pattern Recognition (CVPR) to summarize the most recent contributions and variations to the original algorithm, mostly meant to improve the speed of the algorithm, the robustness and accuracy of the estimated solution and to decrease the dependency from user defined constants.

As pointed out by Torr et al. , RANSAC can be sensitive to the choice of the correct noise threshold that defines which data points fit a model instantiated with a certain set of parameters. If such threshold is too large, then all the hypotheses tend to be ranked equally (good). On the other hand, when the noise threshold is too small, the estimated parameters tend to be unstable (i.e. by simply adding or removing a datum to the set of inliers, the estimate of the parameters may fluctuate). To partially compensate for this undesirable effect, Torr et al. proposed two modification of RANSAC called MSAC (M-estimator SAmple and Consensus) and MLESAC (Maximum Likelihood Estimation SAmple and Consensus).[5] The main idea is to evaluate the quality of the consensus set (i.e. the data that fit a model and a certain set of parameters) calculating its likelihood (whereas in the original formulation by Fischler and Bolles the rank was the cardinality of such set). An extension to MLESAC which takes into account the prior probabilities associated to the input dataset is proposed by Tordoff.[6] The resulting algorithm is dubbed Guided-MLESAC. Along

similar lines, Chum proposed to guide the sampling procedure if some a priori information regarding the input data is known, i.e. whether a datum is likely to be an inlier or an outlier. The proposed approach is called PROSAC, PROgressive SAmple Consensus.[7]

Chum et al. also proposed a randomized version of RANSAC called R-RANSAC [8] to reduce the computational burden to identify a good CS. The basic idea is to initially evaluate the goodness of the currently instantiated model using only a reduced set of points instead of the entire dataset. A sound strategy will tell with high confidence when it is the case to evaluate the fitting of the entire dataset or when the model can be readily discarded. It is reasonable to think that the impact of this approach is more relevant in cases where the percentage of inliers is large. The type of strategy proposed by Chum et al. is called preemption scheme. Nistér proposed a paradigm called Preemptive RANSAC[9] that allows real time robust estimation of the structure of a scene and of the motion of the camera. The core idea of the approach consists in generating a fixed number of hypothesis so that the comparison happens with respect to the quality of the generated hypothesis rather than against some absolute quality metric.

Other researchers tried to cope with difficult situations where the noise scale is not know and/or multiple model instances are present. The first problem has been tackled in the work by Wang and Suter.[10] Toldo et al. represent each datum with the characteristic function of the set of random models that fit the point. Then multiple models are revealed as clusters which group the points supporting the same model. The clustering algorithm, called J-linkage, does not require prior specification of the number of models, nor it necessitate manual parameters tuning.[11]

RANSAC has also been tailored for recursive state estimation applications, where the input measurements are corrupted by outliers and Kalman filter approaches, which rely on a Gaussian distribution of the measurement error, are doomed to fail. Such an approach, it is dubbed KALMANSAC.[12]

6.9 Notes

[1] Data Fitting and Uncertainty, T. Strutz, Springer Vieweg (2nd edition, 2016)

[2] Robust Statistics, Peter. J. Huber, Wiley, 1981 (republished in paperback, 2004), page 1.

[3] Anders Hast, Johan Nysjö, Andrea Marchetti (2013). "Optimal RANSAC – Towards a Repeatable Algorithm for Finding the Optimal Set". Journal of WSCG 21 (1): 21–30.

[4] Hossam Isack, Yuri Boykov (2012). "Energy-based Geometric Multi-Model Fitting". International Journal of Computer Vision 97 (2: 1): 23–147. doi:10.1007/s11263-011-0474-7.

[5] P.H.S. Torr and A. Zisserman, MLESAC: A new robust estimator with application to estimating image geometry, Journal of Computer Vision and Image Understanding 78 (2000), no. 1, 138–156.

[6] B. J. Tordoff and D. W. Murray, Guided-MLESAC: Faster image transform estimation by using matching priors, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005), no. 10, 1523–1535.

[7] Matching with PROSAC - progressive sample consensus, Proceedings of Conference on Computer Vision and Pattern Recognition (San Diego), vol. 1, June 2005, pp. 220–226

[8] O. Chum and J. Matas, Randomized RANSAC with Td,d test, 13th British Machine Vision Conference, September 2002.

[9] D. Nistér, Preemptive RANSAC for live structure and motion estimation, IEEE International Conference on Computer Vision (Nice, France), October 2003, pp. 199–206.

[10] H. Wang and D. Suter, Robust adaptive-scale parametric model estimation for computer vision., IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004), no. 11, 1459-1474

[11] R. Toldo and A. Fusiello, Robust multiple structures estimation with jlinkage, European Conference on Computer Vision (Marseille, France), October 2008, pp. 537–547.

[12] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, KALMANSAC: Robust filtering by consensus, Proceedings of the International Conference on Computer Vision (ICCV), vol. 1, 2005, pp. 633–640

6.10 References

- Martin A. Fischler and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography" (PDF). *Comm. of the ACM* **24** (6): 381–395. doi:10.1145/358669.358692.
- David A. Forsyth and Jean Ponce (2003). *Computer Vision, a modern approach*. Prentice Hall. ISBN 0-13-085198-1.
- Richard Hartley and Andrew Zisserman (2003). *Multiple View Geometry in Computer Vision* (2nd ed.). Cambridge University Press.
- Strutz, T. (2016). *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond)*. 2nd edition, Springer Vieweg. ISBN 978-3-658-11455-8.

- P.H.S. Torr and D.W. Murray (1997). "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix". *International Journal of Computer Vision* **24** (3): 271–300. doi:10.1023/A:1007927408552.
- Ondrej Chum (2005). "Two-View Geometry Estimation by Random Sample and Consensus" (PDF). *PhD Thesis*.
- Sunglok Choi, Taemin Kim, and Wonpil Yu (2009). "Performance Evaluation of RANSAC Family" (PDF). *In Proceedings of the British Machine Vision Conference (BMVC)*.
- Anders Hast, Johan Nysjö, Andrea Marchetti (2013). "Optimal RANSAC – Towards a Repeatable Algorithm for Finding the Optimal Set" (PDF). *Journal of WSCG* **21** (1): 21–30.
- Hossam Isack, Yuri Boykov (2012). "Energy-based Geometric Multi-Model Fitting" (PDF). *International Journal of Computer Vision* **97** (2: 1): 23–147. doi:10.1007/s11263-011-0474-7.

6.11 External links

- OpenCV function: cv::findHomography. OpenCV function: cv::findHomography --- Using RANSAC to find the homography matrix relating two images.
- Fitting ellipses with RANSAC. OpenCV: Fitting ellipses using RANSAC algorithm.
- OpenCV: RANSAC affine transformation. The OpenCV using RANSAC affine transformation estimated estimateAffine2D.
- RANSAC: Homography matrix in Image Stitching. RANSAC is used to estimate the homography matrix in Image Stitching.
- RANSAC Toolbox for MATLAB. A research (and didactic) oriented toolbox to explore the RANSAC algorithm in MATLAB. It is highly configurable and contains the routines to solve a few relevant estimation problems.
- ransac.m The RANSAC algorithm in MATLAB.
- optimalRansac.m The Optimal RANSAC algorithm in MATLAB.
- Implementation in C++ as a generic template.
- Implementation in C++ as a generic template with hyperplane and hypersphere examples.
- RANSAC for Dummies A simple tutorial with many examples that uses the RANSAC Toolbox for MATLAB.
- Source code for RANSAC in MATLAB
- Ransac.js Javascript implementation with visual representation of the iterations (Example of 2D Line fitting).
- ransac.py Python implementation for Scipy/Numpy.
- Scikit-learn and Scikit-image contains Python implementations.
- GML RANSAC Matlab Toolbox – a set of MATLAB scripts, implementing RANSAC algorithm family.
- RANSAC for estimation of geometric transforms - MATLAB examples and help on using RANSAC in Computer Vision applications
- RANSAC Algorithm - Overview of the RANSAC Algorithm
- Random Sample Consensus - Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography
- ACM Digital Library - Library to buy the original article
- PCL - How to use Random Sample Consensus model

6.12 Text and image sources, contributors, and licenses

6.12.1 Text

- **Edge detection** *Source:* https://en.wikipedia.org/wiki/Edge_detection?oldid=700468924 *Contributors:* Michael Hardy, Kku, Cyan, Charles Matthews, Dysprosia, Meduz, Diberri, Giftlite, Seabhcan, BenFrantzDale, Asc99c, Dmmaus, Discospinster, Maye, Forderud, Gengiskanhg, Waldir, RuM, Zbxgscqf, Salix alba, Awotter, Kri, Bgwhite, IBlender, Confuted, Robert L, Jjd27, SmackBot, UrsaFoot, Jcarroll, Kazkaskazkasako, Sct72, MrRadioGuy, Vina-iwbot~enwiki, Acjohnson55, Feraudyh, Yoderj, Hu12, Aursani, Tpl, ChrisCork, Ylloh, CmdrObot, CBM, Eihjia, Cydebot, Alanbly, Kozuch, Medicsl, MistWiz, AntiVandalBot, JAnDbot, David Eppstein, Cspan64, JonMcLoone, SJP, Bonadea, Qtea, Jamelan, Pjoef, Maxlittle2007, Nilx~enwiki, KoenDelaere, Svick, Iknowyourider, Conboy, Binksternet, Mild Bill Hiccup, DumZiBoT, XLinkBot, Manishti2004, Dekart, Ardentbiker, Addbot, Cyborg1158, TutterMouse, Download, LaaknorBot, 5 albert square, Křžut, Willondon, Yobot, AnomieBOT, ArthurBot, Shcha, Dneprodzerzhinsk, Weichaoliu, MuffledThud, FrescoBot, Ahauptfleisch, Geazzo, Kwiki, Pinethicket, North8000, Clarkcj12, Frank.vanMeurs, Mskayali, K6ka, ZéroBot, Amogh mahapatra, Makecat, FinalRapture, Jtherr, ClueBot NG, Frietjes, Roller958, BG19bot, MusikAnimal, BattyBot, Sai123k, Kolega2357, Mark viking, Tentinator, Samratsubedi, AndyThe, The Code Blitzer, Ahmedskiko, Uqam, Edge detection, Mullder19731973, Davidstone1415 and Anonymous: 110
- **Dilation (morphology)** *Source:* https://en.wikipedia.org/wiki/Dilation_(morphology)?oldid=690313578 *Contributors:* DJ Clayworth, Robbot, Seabhcan, Blankfaze, Yann.gavet, Woohookitty, CmdrObot, Speedyboy, User A1, R'n'B, LokiClock, Wiae, Mpd1989, Niceguyedc, Stlman, Qwfp, Dboehmer, Addbot, Mortense, Renatokeshet, Adelpine, Miym, Omnipaedista, DrilBot, Bracchesimo, Hommemodels, EmausBot, ClueBot NG, Helpful Pixie Bot and Anonymous: 18
- **Blob detection** *Source:* https://en.wikipedia.org/wiki/Blob_detection?oldid=699882882 *Contributors:* Fjarlq, Rich Farmbrough, Rjwilmsi, Petter Strandmark, SmackBot, Silly rabbit, Euchiasmus, Rigadoun, Agent007bond, Tpl, Amedlin, Cydebot, Casmith 789, JNW, Rich257, Jamesontai, Monty845, Hairybrute, 1ForTheMoney, Dekart, Addbot, DOI bot, Cbauckhage, Yobot, Sorry40times, Qorilla, Songuke, Citation bot 1, Rbteye, Trappist the monk, Salmarina, Kenchikuben, RjwilmsiBot, Dewritech, Tommy2010, Coupriec, Mentibot, Snotbot, Helpful Pixie Bot, Junslife, Happyuk, JYBot, Lone boatman, Inigobar, Mark viking, Kix.swamp, Selisel88 and Anonymous: 40
- **Hough transform** *Source:* https://en.wikipedia.org/wiki/Hough_transform?oldid=705615786 *Contributors:* Damian Yerrick, Kku, Komap, IMSoP, Dysprosia, Ashwin, Marius~enwiki, Seabhcan, BenFrantzDale, CryptoDerk, MarkSweep, AndrewKeenanRichardson, Rich Farmbrough, Billlion, Kwamikagami, Hart~enwiki, Don Reba, Flambe, Yurivict, Forderud, Japanese Searobin, Angr, TheGoblin, Tevatron~enwiki, Mandarax, BD2412, Rjwilmsi, Ckelloug, Erkcan, MikeJ9919, FlaBot, RobertG, Wavelength, Edouard-lopez, Mike1024, Arunvijayan, Zvika, Jjd27, SmackBot, Not cat, Unyoyega, KYN, Ohnoitsjamie, AndrewKay, Bluebot, Thumperward, OrphanBot, Slogan621, Disavian, Feraudyh, Ziplux~enwiki, Tpl, Cydebot, Thijs!bot, Deflective, Stangaa, Vernanimalcula, Brianvon, Glrx, Dispenser, Haseldon, Sgeureka, Akhram, VolkovBot, TXiKiBoT, Crohnie, Lgrove, Nguyener, Bethjaneway, Illuminated, Kevin.mcguinness, Herry13, KoenDelaere, Svick, Enfenion, Iknowyourider, ClueBot, Justin W Smith, Alexbot, Sowmyar, Jwpat7, Qwfp, DumZiBoT, Davis685, Dthomsen8, Dekart, Sameer0s, Addbot, Mortense, MrOllie, SpBot, AgadaUrbanit, Cdmay, Lightbot, Luckas-bot, Yobot, AnomieBOT, Ciphers, 1exec1, Laffernandes, Shcha, The Evil IP address, Control.valve, Tahirrabbani, Bunnyyyy, Editwikiftw, Tetraedycal, PrincessofLlyr, Potentials, Felis domestica, Bmitov, JosephCatrambone, Eigma, Zafar142003, Steffenwolf~enwiki, Nachiket4you, Francois Boulogne, Lyla1205, Helpful Pixie Bot, Amndeep7, 1w2w3y, Alb d91, EdwinDelgadoH, Dexbot, Codeatlas2112, Archmarch, Maimonid, AndyThe, Fredericoal, Utopcu, Hossameldeenfci and Anonymous: 135
- **Circle Hough Transform** *Source:* https://en.wikipedia.org/wiki/Circle_Hough_Transform?oldid=667639775 *Contributors:* RJFJR, JHCaufield, Yobot, 1w2w3y and Anonymous: 2
- **RANSAC** *Source:* https://en.wikipedia.org/wiki/RANSAC?oldid=702232986 *Contributors:* Fnielsen, Kku, Kosebamse, Mark Foskey, MathMartin, Msm, Simoneau, Andreas Kaufmann, Richie, Nowozin, Daniel Case, GregorB, Qwertyus, Rjwilmsi, Adoniscik, Cholmes75, SmackBot, KYN, Bluebot, Hongooi, Ajheller, Eigensoul, Disavian, Kjkjava, David Cooke, Thijs!bot, JasonCrawford, Martyr2566, Dogaroon, Ilion2, Lovibond, Dontdoit, Joshua Issac, Semifinalist, Lourakis, Melcombe, PerryTachett, Avenged Eightfold, PixelBot, Addbot, SupperTina, Ettrig, Luckas-bot, AnomieBOT, Citation bot, Xqbot, HannesP, Lucasmus, Citation bot 1, Compvis, Diannaa, Ripchip Bot, Sweot.ttam, EmausBot, John of Reading, ARuslanova, ClueBot NG, KLBot2, ChrisGualtieri, Dexbot, SnippyHolloW, LokeshRavindranathan, Densonsmith, Avi.nehemiah, The Code Blitzer, Monkbot, Xavi.borras, Brian moore81, Stephenweixu, Epsiloneternal and Anonymous: 57

6.12.2 Images

- **File:Circle_Hough_transform_of_four_points_on_3_circles.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/67/Circle_Hough_transform_of_four_points_on_3_circles.jpg *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* 1w2w3y
- **File:Circle_Hough_transform_of_four_points_on_a_circle.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/cc/Circle_Hough_transform_of_four_points_on_a_circle.jpg *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* 1w2w3y
- **File:Dilation.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/8d/Dilation.png *License:* Public domain *Contributors:* Transferred from en.wikipedia to Commons by Adelpine using CommonsHelper. *Original artist:* Renatokeshet at English Wikipedia
- **File:EdgeDetectionMathematica.png** *Source:* https://upload.wikimedia.org/wikipedia/en/8/8e/EdgeDetectionMathematica.png *License:* CC-BY-SA-3.0 *Contributors:*

  I (JonMcLoone (talk)) created this work entirely by myself. *Original artist:*

  JonMcLoone (talk)
- **File:Edit-clear.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg *License:* Public domain *Contributors:* The *Tango! Desktop Project*. *Original artist:*

  The people from the Tango! project. And according to the meta-data in the file, specifically: "Andreas Nilsson, and Jakub Steiner (although minimally)."
- **File:Find_circles_in_shoeprint.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/46/Find_circles_in_shoeprint.jpg *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* 1w2w3y

- **File:Hough-example-result-en.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1c/Hough-example-result-en.png *License:* CC BY 2.5 *Contributors:* Own work *Original artist:* Daf-de
- **File:Hough_space_plot_example.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/af/Hough_space_plot_example.png *License:* Public domain *Contributors:* the English language Wikipedia (log) *Original artist:* Mike1024
- **File:Hough_transform_diagram.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/39/Hough_transform_diagram.png *License:* Public domain *Contributors:* http://en.wikipedia.org/wiki/Image:Hough_transform_diagram.png *Original artist:* Mike1024
- **File:PST_edge_detector_saint_Paul.tif** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/c0/PST_edge_detector_saint_Paul.tif *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Uqam
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*

  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:* Tkgd2007
- **File:R_theta_line.GIF** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/e6/R_theta_line.GIF *License:* Public domain *Contributors:* Own work (Original text: *I (Shcha (talk)) created this work entirely by myself.*) *Original artist:* Shcha (talk)

6.12.3 Content license

- Creative Commons Attribution-Share Alike 3.0