



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export

Create a book
Download as PDF
Printable version

Languages

Čeština
Deutsch
Español
Français
Italiano
עברית
Latina
Nederlands
Polski
Português
Русский
Українська
中文

Edit links

Create account Log in

Article Talk

Read

Edit

More▼

Search



Canny edge detector

From Wikipedia, the free encyclopedia

The **Canny edge detector** is an [edge detection](#) operator that uses a multi-stage [algorithm](#) to detect a wide range of edges in images. It was developed by [John F. Canny](#) in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.

Contents [hide]

- Development of the Canny algorithm
- Process of Canny edge detection algorithm
 - Gaussian Filter
 - Finding the Intensity Gradient of the Image
 - Non-maximum Suppression
 - Double Threshold
 - Edge Tracking by Hysteresis
- Improvement on Canny Edge Detection
 - Replace Gaussian Filter
 - Improvement on gradient magnitude and direction calculation
 - Robust method to determine the dual-threshold value
 - The thinning of the edge
- Differential geometric formulation of the Canny edge detector
- Variational formulation of the Haralick–Canny edge detector
- Parameters
- Conclusion
- See also
- References
- External links

Feature detection

Edge detection

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

Corner detection

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

Blob detection

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

Ridge detection

Hough transform

Structure tensor

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

Feature description

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

Scale space

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

v · t · e

Development of the Canny algorithm [edit]

Edge detection, especially step edge detection has been widely applied in various computer vision systems, which is an important technique to extract useful

structural information from different vision objects and dramatically reduce the amount of data to be processed. Canny has found that, the requirements for the application of edge detection on diverse vision systems are relatively the same. Thus, a development of an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection includes

- Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
- The edge point detected from the operator should accurately localize on the center of the edge.
- a given edge in the image should only be marked once, and where possible, image noise should not create false edges.

To satisfy these requirements Canny used the [calculus of variations](#) – a technique which finds the [function](#) which optimizes a given [functional](#). The optimal function in Canny's detector is described by the sum of four [exponential](#) terms, but it can be approximated by the first [derivative](#) of a [Gaussian](#).

Among the edge detection methods developed so far, canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it becomes one of the most popular algorithms for edge detection.

Process of Canny edge detection algorithm [edit]

The Process of Canny edge detection algorithm can be broken down to 5 different steps:



The Canny edge detector applied to a colour photograph of a steam engine.



The original image.

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Every step will be described in details as following. The introduction of procedure below is developed based on Prof Thomas Moeslund's lecture note for digital image processing in Indian Institute of Technology.[5]

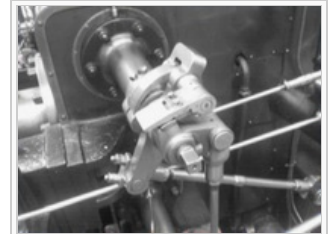
Gaussian Filter [\[edit\]](#)

Since all edge detection results are easily affected by image noise, it is essential to filter out the noise to prevent false detection caused by noise. To smooth the image, a Gaussian filter is applied to convolve with the image. This step will slightly smooth the image to reduce the effects of obvious noise on the edge detector. The equation for a Gaussian filter kernel of size $(2k+1) \times (2k+1)$ is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-k-1)^2 + (j-k-1)^2}{2\sigma^2}\right)$$

Here is an example of a 5×5 Gaussian filter, used to create the image to the right, with $\sigma = 1.3$. (The asterisk denotes a [convolution](#) operation.)

$$\mathbf{B} = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$



The image after a 5×5 Gaussian mask has been passed across each pixel.

It is important to understand that the selection of the size of the Gaussian kernel will affect the performance of the detector. The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases, but this will also vary depending on specific situations.

Finding the Intensity Gradient of the Image [\[edit\]](#)

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The [edge detection operator](#) ([Roberts](#), [Prewitt](#), [Sobel](#) for example) returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient and direction can be determined:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\Theta = \text{atan2}(G_y, G_x).$$

where G can be computed using the [hypot](#) function and [atan2](#) is the arctangent function with two arguments. The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0° , 45° , 90° and 135° for example). An edge direction falling in each color region will be set to a specific angle values, for example alpha lying in yellow region (0° to 22.5° and 157.5° to 180°) will be set to 0° .

Non-maximum Suppression [\[edit\]](#)

Non-maximum suppression is an [edge thinning](#) technique.

Non-Maximum suppression is applied to "thin" the edge. After applying gradient calculation, the edge extracted from the gradient value is still quite blurred. With respect to criteria 3, there should only be one accurate response to the edge. Thus non-maximum suppression can help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:

1. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction(i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. For example,

- if the rounded gradient angle is 0° (i.e. the edge is in the north–south direction) the point will be considered to be on

- the edge if its gradient magnitude is greater than the magnitudes at pixels in the **east and west** directions,
- if the rounded gradient angle is 90° (i.e. the edge is in the east–west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north and south** directions,
- if the rounded gradient angle is 135° (i.e. the edge is in the northeast–southwest direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north west and south east** directions,
- if the rounded gradient angle is 45° (i.e. the edge is in the north west–south east direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the **north east and south west** directions.

In more accurate implementations, linear interpolation is used between the two neighbouring pixels that straddle the gradient direction. For example, if the gradient angle is between 45° and 90° , interpolation between gradients at the **north** and **north east** pixels will give one interpolated value, and interpolation between the **south** and **south west** pixels will give the other (using the conventions of last paragraph). The gradient magnitude at the central pixel must be greater than both of these for it to be marked as an edge.

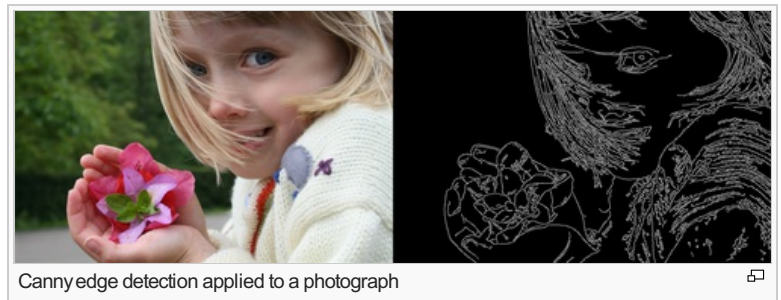
Note that the sign of the direction is irrelevant, i.e. north–south is the same as south–north and so on.

Double Threshold [\[edit\]](#)

After application of non-maximum suppression, the edge pixels are quite accurate to present the real edge. However, there are still some edge pixels at this point caused by noise and color variation. In order to get rid of the spurious responses from these bothering factors, it is essential to filter out the edge pixel with the weak gradient value and preserve the edge with the high gradient value. Thus two threshold values are set to clarify the different types of edge pixels, one is called high threshold value and the other is called the low threshold value. If the edge pixel's gradient value is higher than the high threshold value, they are marked as strong edge pixels. If the edge pixel's gradient value is smaller than the high threshold value and larger than the low threshold value, they are marked as weak edge pixels. If the pixel value is smaller than the low threshold value, they will be suppressed. The two threshold values are empirically determined values, which will need to be defined when applying to different images.

Edge Tracking by Hysteresis [\[edit\]](#)

So far, the strong edge pixels should certainly be involved in the final edge image, as they are extracted from the true edges in the image. However, there will be some debate on the weak edge pixels, as these pixels can either be extracted from the true edge, or the noise/color variations. To achieve an accurate result, the weak edges



caused from the latter reasons should be removed. The criteria to determine which case does the weak edge belongs to is that, usually the weak edge pixel caused from true edges will be connected to the strong edge pixel. To track the edge connection, Binary Large Object-analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels. As long as there is one strong edge pixel is involved in the BLOB, that weak edge point can be identified as one that should be preserved.

Improvement on Canny Edge Detection [\[edit\]](#)

While traditional canny edge detection provides relatively simple but precise methodology for edge detection problem, with the more demanding requirements on the accuracy and robustness on the detection, the traditional algorithm can no longer handle the challenging edge detection task. The main defects of the traditional algorithm can be summarized as following:[8]

1. Gaussian filter is applied to smooth out the noise, but it will also smooth the edge, which is considered as the high frequency feature. This will increase the possibility to miss weak edges, and the appearance of isolated edges in the result.
2. For the gradient amplitude calculation, the old canny edge detection algorithm uses center in a small 2×2 neighborhoods window to calculate the finite difference mean value to represent the gradient amplitude. This method is sensitive to noise and can easily detect fake edges and lose real edges.
3. In traditional canny edge detection algorithm, there will be two fixed global threshold values to filter out the false edges. However, as the image gets complex, different local areas will need very different threshold values to accurately find the real edges. In addition, the global threshold values are determined manually through experiments in the traditional method, which leads to complexity of calculation when large number of different images needs to be dealt with.
4. The result of the traditional detection cannot reach a satisfactory high accuracy of single response for each

edge- multi-point responses will appear.

In order to address these defects, improvement for the canny edge algorithm is added in the fields below.

Replace Gaussian Filter [\[edit\]](#)

As both edge and noise will be identified as high frequency signal, simple Gaussian filter will add smooth effect on both of them. However, in order to reach high accuracy of detection of the real edge, it is expected that more smooth effect should be added to noise and less smooth effect should be added to the edge. Bing Wang and Shaosheng Fan from Changsha University of Science and Technology developed an adaptive filter, where the filter will evaluate discontinuity between greyscale values of each pixel.[8] The higher the discontinuity, the lower the weight value is set for the smooth filter at that point. Contrarily, the lower the discontinuity between the greyscale values, the higher the weight value is set to the filter. The process to implement this adaptive filter can be summarized in five step:

1. $K = 1$, set the iteration n and the coefficient of the amplitude of the edge h .
2. Calculate the gradient value $G_x(x, y)$ and $G_y(x, y)$
3. Calculate the weight according to the formula below:

$$d(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$$

$$w(x, y) = \exp\left(-\frac{\sqrt{d(x, y)}}{2h^2}\right)$$

4. The definition of the adaptive filter is:

$$f(x, y) = \frac{1}{N} \sum_{i=-1}^1 \sum_{j=-1}^1 f(x+i, y+j) w(x+i, y+j)$$

to smooth the image, which

$$N = \sum_{i=-1}^1 \sum_{j=-1}^1 w(x+i, y+j)$$

5. When $K = n$, stop the iterative, otherwise, $k = k+1$, keep do the second step

Improvement on gradient magnitude and direction calculation [\[edit\]](#)

The study work conducted by Ping Zhou and his colleagues resolve the high sensitivity's issue brought by the small-scale window to calculate the gradient magnitude values and directions.[10] Instead of using a 2×2 neighborhood window to calculate the gradient magnitude values and directions, Ping developed a 3×3 neighborhood windows to calculate the gradient values, so that a better magnitude and direction value can be calculated. The equations are demonstrated as following:

$$G_x(x, y) = [I(i, j+1) - I(i, j-1) + I(i-1, j+1) - I(i-1, j-1) + I(i+1, j+1) - I(i+1, j-1)]/2$$
$$G_y(x, y) = [I(i+1, j) - I(i-1, j) + I(i+1, j-1) - I(i-1, j-1) + I(i+1, j+1) - I(i-1, j+1)]/2$$

Robust method to determine the dual-threshold value [\[edit\]](#)

In order to resolve the challenges where it is hard to determine the dual-threshold value empirically, a Japanese scholar, Otsu [11] has developed an adaptive method to determine the threshold value for different images. What he does is that he has put all the pixel values in the image into two groups c_0 and c_1 , which is separated by an unknown threshold value T . Defining the corresponding pixel number of intensity level i is noted as n_i , thus the probability is defined as:

$$p_i = \frac{n_i}{n}$$

Where n is the total number of the pixel points in the image The mean value of the gray level distribution probability is defined as :

$$u_T = \sum_{i=0}^{L-1} \frac{i p_i}{w_0}$$
$$w_0 = \sum_{i=0}^{L-1} p_i$$
$$u_1 = \sum_{i=T+1}^{L-1} \frac{i p_i}{w_1}$$
$$w_1 = 1 - w_0$$

Between-class variance is defined as: The high threshold value T is determined as the one that can maximize the value of σ_b^2 and the lower threshold value is determined as $Tl = 0.5 * Th$. In this way, for each image, an adaptive dual threshold value can be best determined to filter out the pixel values that are not considered as edges.

The thinning of the edge [\[edit\]](#)

While the traditional canny edge detection have implemented a good detection result to meet with the first two criteria, it does not meet with the single response per edge strictly. A mathematical morphology to thin the detected edge is developed by Mallat S and Zhong.[9]

Differential geometric formulation of the Canny edge detector [\[edit\]](#)

A more refined approach to obtain edges with sub-pixel accuracy is by using the approach of [differential edge detection](#), where the requirement of non-maximum suppression is formulated in terms of second- and third-order derivatives computed from a [scale space](#) representation (Lindeberg 1998) – see the article on [edge detection](#) for a detailed description.

Variational formulation of the Haralick–Canny edge detector [\[edit\]](#)

A variational explanation for the main ingredient of the Canny edge detector, that is, finding the zero crossings of the 2nd derivative along the gradient direction, was shown to be the result of minimizing a Kronrod–Minkowski functional while maximizing the integral over the alignment of the edge with the gradient field (Kimmel and Bruckstein 2003). See article on regularized Laplacian zero crossings and other optimal edge integrators for a detailed description.

Parameters [\[edit\]](#)

The Canny algorithm contains a number of adjustable parameters, which can affect the computation time and effectiveness of the algorithm.

- The size of the Gaussian filter: the smoothing filter used in the first stage directly affects the results of the Canny algorithm. Smaller filters cause less blurring, and allow detection of small, sharp lines. A larger filter causes more blurring, smearing out the value of a given pixel over a larger area of the image. Larger blurring radii are more useful for detecting larger, smoother edges – for instance, the edge of a rainbow.
- Thresholds: the use of two thresholds with hysteresis allows more flexibility than in a single-threshold approach, but general problems of thresholding approaches still apply. A threshold set too high can miss important information. On the other hand, a threshold set too low will falsely identify irrelevant information (such as noise) as important. It is difficult to give a generic threshold that works well on all images. No tried and tested approach to this problem yet exists.

Conclusion [\[edit\]](#)

The Canny algorithm is adaptable to various environments. Its parameters allow it to be tailored to recognition of edges of differing characteristics depending on the particular requirements of a given implementation. In Canny's original paper, the derivation of the optimal filter led to a [Finite Impulse Response](#) filter, which can be slow to compute in the spatial domain if the amount of smoothing required is important (the filter will have a large spatial support in that case). For this reason, it is often suggested to use Rachid Deriche's [infinite impulse response](#) form of Canny's filter (the [Canny–Deriche detector](#)), which is recursive, and which can be computed in a short, fixed amount of time for any desired amount of smoothing. The second form is suitable for real time implementations in [FPGAs](#) or [DSPs](#), or very fast embedded PCs. In this context, however, the regular recursive implementation of the Canny operator does not give a good approximation of rotational symmetry and therefore gives a bias towards horizontal and vertical edges.



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

- [Feature detection \(computer vision\)](#)
- [Feature extraction](#)
- [Scale space](#)
- [Ridge detection](#)
- [Computer vision](#)
- [Digital image processing](#)















References [\[edit\]](#)

1. Canny, J., *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.
2. R. Deriche, *Using Canny's criteria to derive a recursively implemented optimal edge detector*, Int. J. Computer Vision, Vol. 1, pp. 167–187, April 1987.
3. Lindeberg, Tony "Edge detection and ridge detection with automatic scale selection", *International Journal of Computer Vision*, 30, 2, pp 117—154, 1998. (Includes the differential approach to non-maximum suppression.) [↗](#)
4. Kimmel, Ron and Bruckstein, Alfred M. "On regularized Laplacian zero crossings and other optimal edge integrators", *International Journal of Computer Vision*, 53(3):225–243, 2003. (Includes the geometric variational

interpretation for the Haralick–Canny edge detector.) 

5. Moeslund, T. (2009, March 23). Canny Edge Detection. Retrieved December 3, 2014 
6. Thomas B. Moeslund. Image and Video Processing. August 2008
7. Green, B. (2002, January 1). Canny Edge Detection Tutorial. Retrieved December 3, 2014 
8. Li, Q., Wang, B., & Fan, S. (2009). Browse Conference Publications Computer Science and Engineer ... Help Working with Abstracts An Improved CANNY Edge Detection Algorithm. In 2009 Second International Workshop on Computer Science and Engineering proceedings : WCSE 2009 : 28–30 October 2009, Qingdao, China (pp. 497–500). Los Alamitos, CA: IEEE Computer Society
9. Mallat S, Zhong S. Characterization of Signals from Multi scale Edges [J]. IEEE Trans on PAMI, 1992, 14 (7):710-732.
10. Zhou, P., Ye, W., & Wang, Q. (2011). An Improved Canny Algorithm for Edge Detection. Journal of Computational Information Systems, 7(5), 1516-1523.
11. Otsu N. A threshold selection method from gray-level histograms. IEEE Trans Systems, Man and Cybernetics,9(1):62-66,1979.

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

- [John Canny's home page](#) 
- [Publication List of Rachid Deriche](#) 
- [Journal Publications of Ron Kimmel](#) 
- [Easy-to-follow MIT licensed c implementation](#) 
- [Canny edge detection in c++ OpenCV](#) 
- [Canny edge detection in Python OpenCV](#) 
- [Free Java implementation of Canny edge detector](#) 
- [Canny edge detector in Mathematica](#) 
- [Edge detection in MATLAB](#) 
- [Canny edge detector implementation in ActionScript for the Flash Platform](#) 
- [On-line Canny edge detector](#) 
- [Matlab Code for Canny Edge Detection from UC Berkeley](#) 
- [File Exchange for Canny Edge Detection Code](#) 
- [Canny Edge Detection in Java](#) 

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

This page was last modified on 30 August 2015, at 10:15.

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

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

