Purpose

Find edges in an intensity image

Syntax

```
BW = edge(I,'sobel')
BW = edge(I,'sobel',thresh)
BW = edge(I,'sobel',thresh,direction)
[BW,thresh] = edge(I,'sobel',...)

BW = edge(I,'prewitt')
BW = edge(I,'prewitt',thresh)
BW = edge(I,'prewitt',thresh,direction)
[BW,thresh] = edge(I,'prewitt',...)

BW = edge(I,'roberts')
BW = edge(I,'roberts',thresh)
[BW,thresh] = edge(I,'roberts',...)

BW = edge(I,'log')
BW = edge(I,'log',thresh)
BW = edge(I,'log',thresh,sigma)
[BW,threshold] = edge(I,'log',...)

BW = edge(I,'zerocross',thresh,h)
[BW,thresh] = edge(I,'zerocross',...)

BW = edge(I,'canny')
BW = edge(I,'canny',thresh)
BW = edge(I,'canny',thresh,sigma)
[BW,threshold] = edge(I,'canny',...)
```

Description

`edge` takes an intensity image `I` as its input, and returns a binary image `BW` of the same size as `I`, with 1's where the function finds edges in `I` and 0's elsewhere.

edge supports six different edge-finding methods:

- The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.
- The zero-cross method finds edges by looking for zero crossings after filtering I with a filter you specify.
- The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges.

The parameters you can supply differ depending on the method you specify. If you do not specify a method, edge uses the Sobel method.

Sobel Method

`BW = edge(I,'sobel')` specifies the Sobel method.

`BW = edge(I,'sobel',thresh)` specifies the sensitivity threshold for the Sobel method. edge ignores all edges that are not stronger than `thresh`. If you do not specify `thresh`, or if `thresh` is empty ([]), edge chooses the value automatically.

`BW = edge(I,'sobel',thresh,direction)` specifies directionality for the Sobel method. `direction` is a string specifying whether to look for `'horizontal'` or `'vertical'` edges, or `'both'` (the default).

`[BW,thresh] = edge(I,'sobel',...)` returns the threshold value.

Prewitt Method

`BW = edge(I,'prewitt')` specifies the Prewitt method.

`BW = edge(I,'prewitt',thresh)` specifies the sensitivity threshold for the Prewitt method. `edge` ignores all edges that are not stronger than `thresh`. If you do not specify `thresh`, or if `thresh` is empty (`[]`), `edge` chooses the value automatically.

`BW = edge(I,'prewitt',thresh,direction)` specifies directionality for the Prewitt method. `direction` is a string specifying whether to look for `'horizontal'` or `'vertical'` edges, or `'both'` (the default).

`[BW,thresh] = edge(I,'prewitt',...)` returns the threshold value.

Roberts Method

`BW = edge(I,method)` specifies the Roberts method.

`BW = edge(I,method,thresh)` specifies the sensitivity threshold for the Roberts method. `edge` ignores all edges that are not stronger than `thresh`. If you do not specify `thresh`, or if `thresh` is empty (`[]`), `edge` chooses the value automatically.

`[BW,thresh] = edge(I,method,...)` returns the threshold value.

Laplacian of Gaussian Method

`BW = edge(I,'log')` specifies the Laplacian of Gaussian method.

`BW = edge(I,'log',thresh)` specifies the sensitivity threshold for the Laplacian of Gaussian method. `edge` ignores all edges that are not stronger than `thresh`. If you do not specify `thresh`, or if `thresh` is empty (`[]`), `edge` chooses the value automatically.

`BW = edge(I,'log',thresh,sigma)` specifies the Laplacian of Gaussian method, using `sigma` as the standard deviation of the LoG filter. The default `sigma` is 2; the size of the filter is n-by-n, where `n = ceil(sigma*3)*2+1`.

`[BW,thresh] = edge(I,'log',...)` returns the threshold value.

Zero-cross Method

`BW = edge(I,'zerocross',thresh,h)` specifies the zero-cross method, using the filter `h`. `thresh` is the sensitivity threshold; if the argument is empty ([]), `edge` chooses the sensitivity threshold automatically.

`[BW,thresh] = edge(I,'zerocross',...)` returns the threshold value.

Canny Method

`BW = edge(I,'canny')` specifies the Canny method.

`BW = edge(I,'canny',thresh)` specifies sensitivity thresholds for the Canny method. `thresh` is a two-element vector in which the first element is the low threshold, and the second element is the high threshold. If you specify a scalar for `thresh`, this value is used for the high threshold and `0.4*thresh` is used for the low threshold. If you do not specify `thresh`, or if `thresh` is empty ([]), `edge` chooses low and high values automatically.

`BW = edge(I,'canny',thresh,sigma)` specifies the Canny method, using `sigma` as the standard deviation of the Gaussian filter. The default `sigma` is 1; the size of the filter is chosen automatically, based on `sigma`.

`[BW,thresh] = edge(I,'canny',...)` returns the threshold values as a two-element vector.

Class Support

`I` can be of class `uint8` or `double`. `BW` is of class `uint8`.

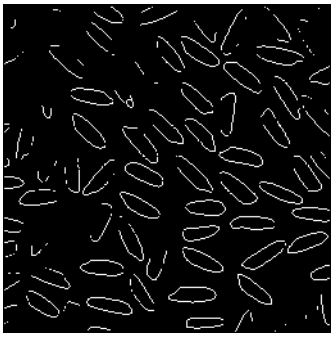
Remarks

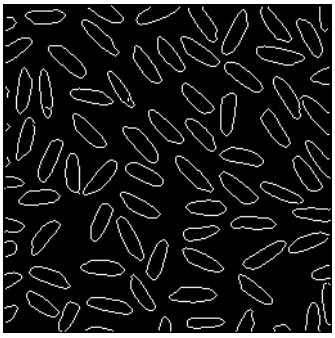
For the `'log'` and `'zerocross'` methods, if you specify a threshold of 0, the output image has closed contours, because it includes all of the zero crossings in the input image.

Example

Find the edges of the `rice.tif` image using the Prewitt and Canny methods:

```
I = imread('rice.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
imshow(BW1)
figure, imshow(BW2)
```





References

Lim, Jae S. *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1990. pp. 478-488.

Parker, James R. *Algorithms for Image Processing and Computer Vision.* New York: John Wiley & Sons, Inc., 1997. pp. 23-29.

Canny, John. "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1986. Vol. PAMI-8, No. 6, pp. 679-698.