Edge Detection

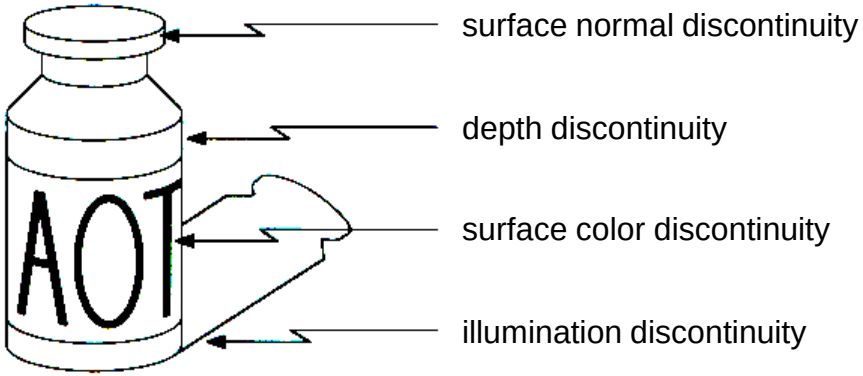
Many slides from Lana Lazebnik, Steve Seitz, David Forsyth, David Lowe, Fei-Fei Li, and Derek Hoiem

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

Source: D. Lowe

Origin of Edges



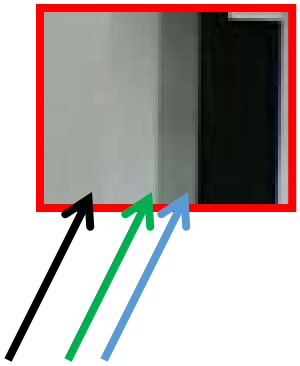
- Edges are caused by a variety of factors

Source: Steve Seitz

Closeup of edges

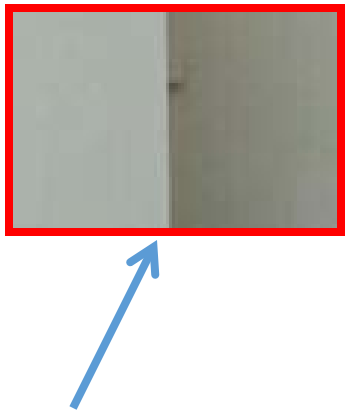
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges

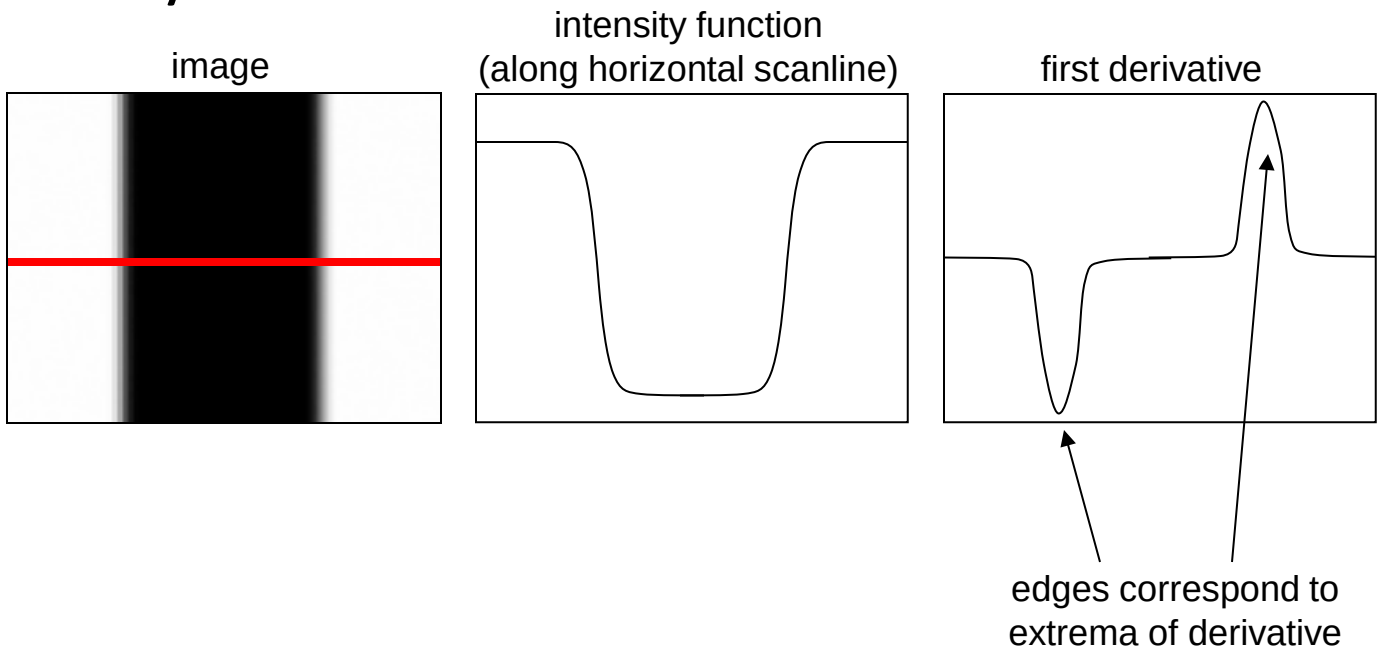
Source: D. Hoiem

Edge detection

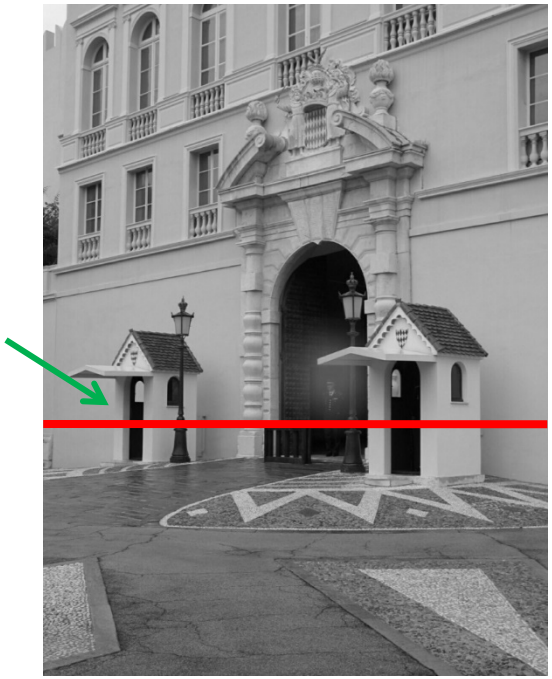
- Edges are places in the image with strong intensity contrast.
- Since edges often occur at image locations representing object boundaries.
- Edge detection is extensively used in **image segmentation** when we want to divide the image into areas corresponding to different objects.
- Representing an image by its edges has the further advantage that the **amount of data is reduced** significantly while retaining most of the image information.
- Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a highpass **frequency filter** in the Fourier domain or by **convolving** the image with an appropriate **kernel** in the spatial domain.
- In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results.

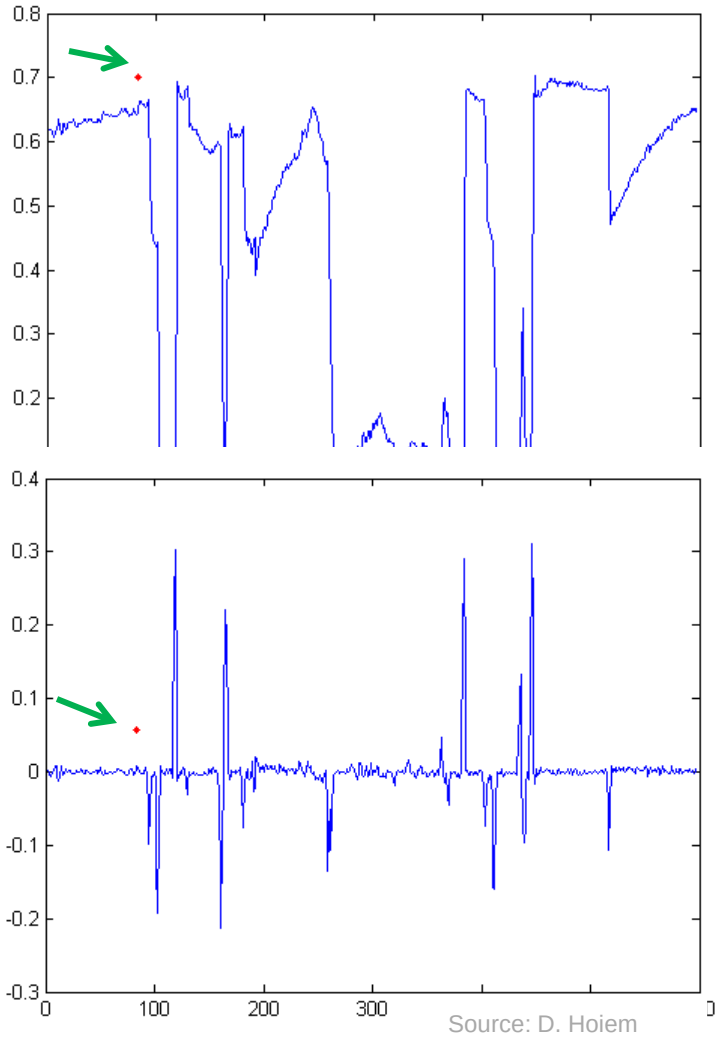
Characterizing edges

- An edge is a place of rapid change in the image intensity function



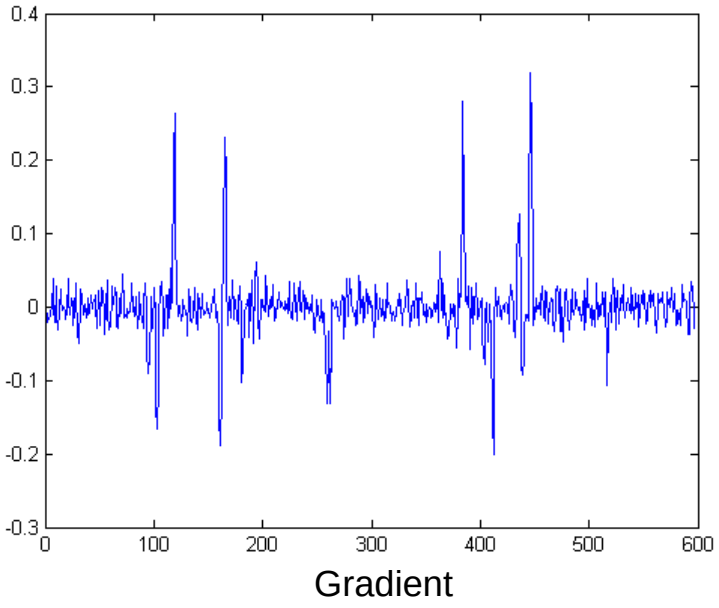
Intensity profile





Source: D. Hoiem

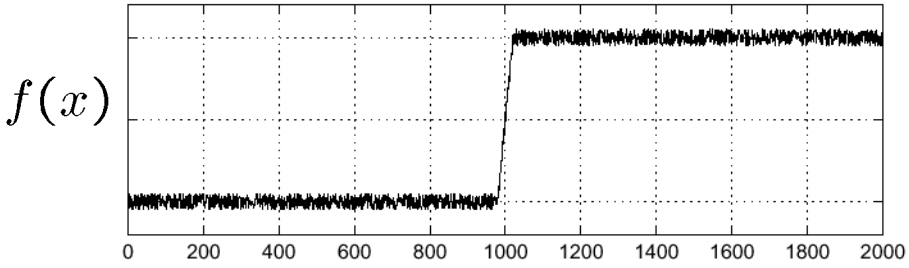
With a little Gaussian noise

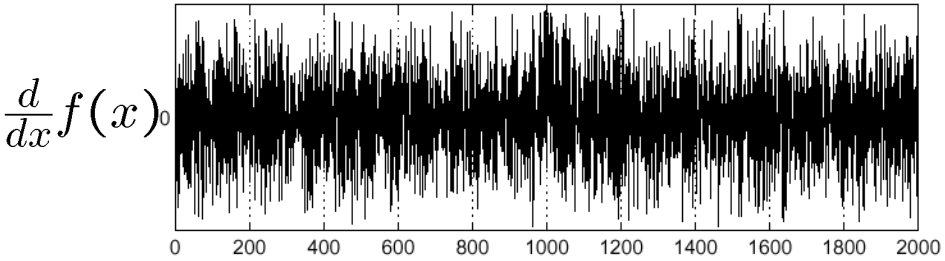


Source: D. Hoiem

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

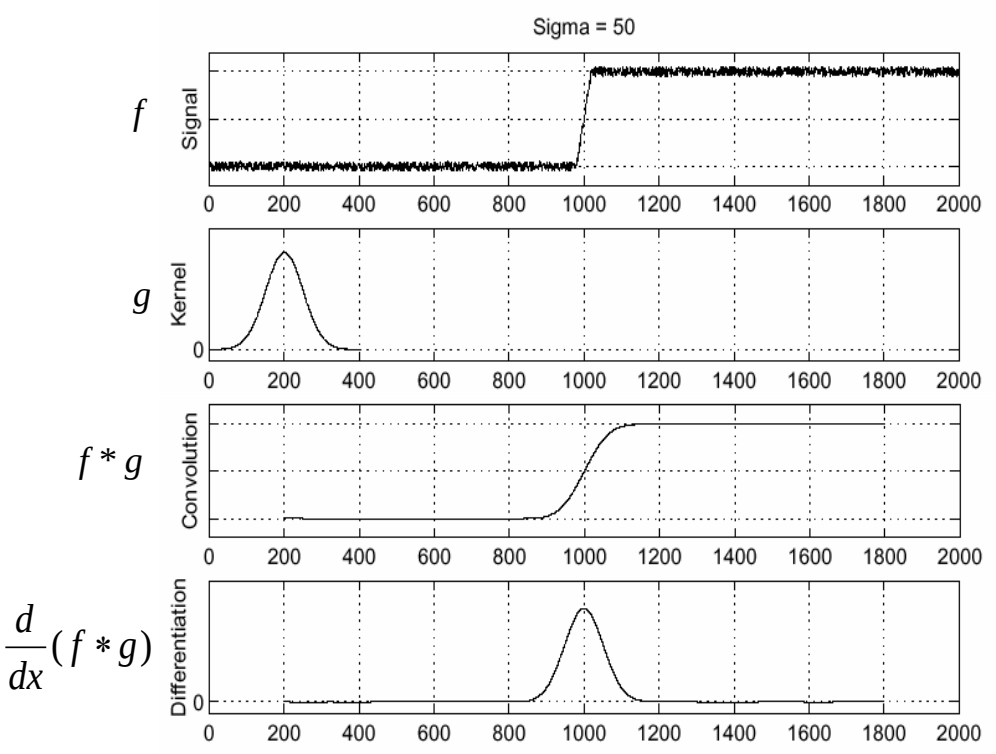
Source: S. Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

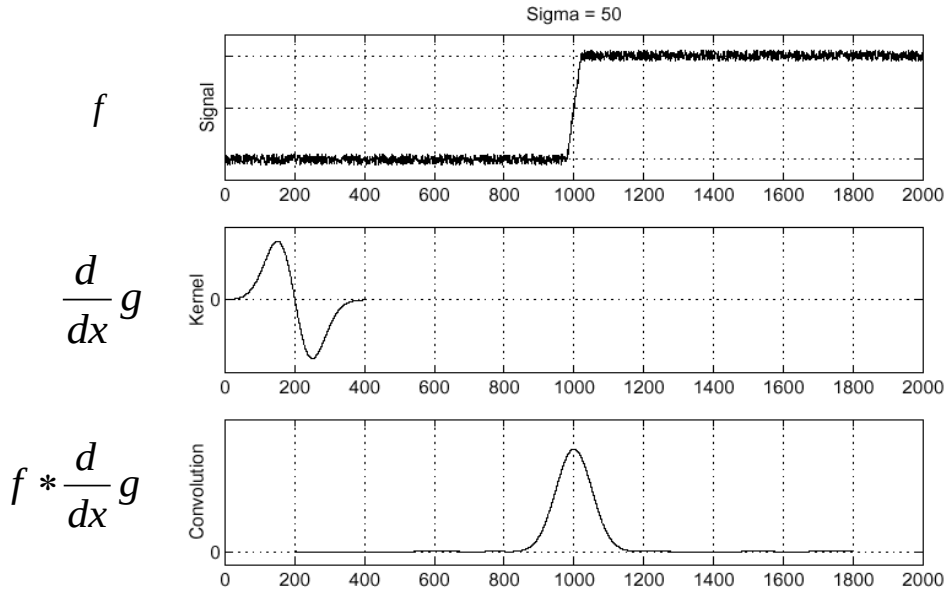
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

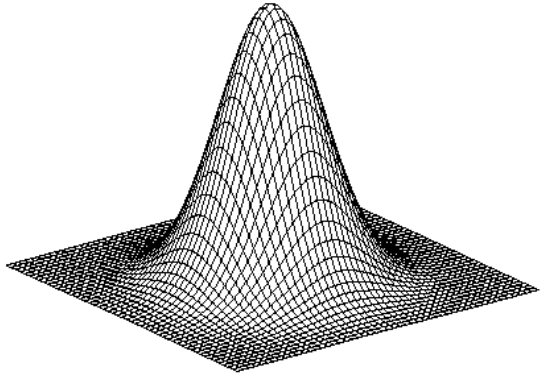
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

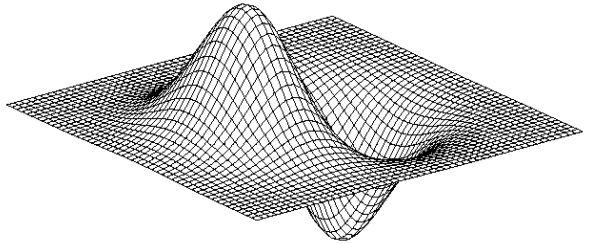


Source: S. Seitz

Derivative of Gaussian filter



* [1 -1] =



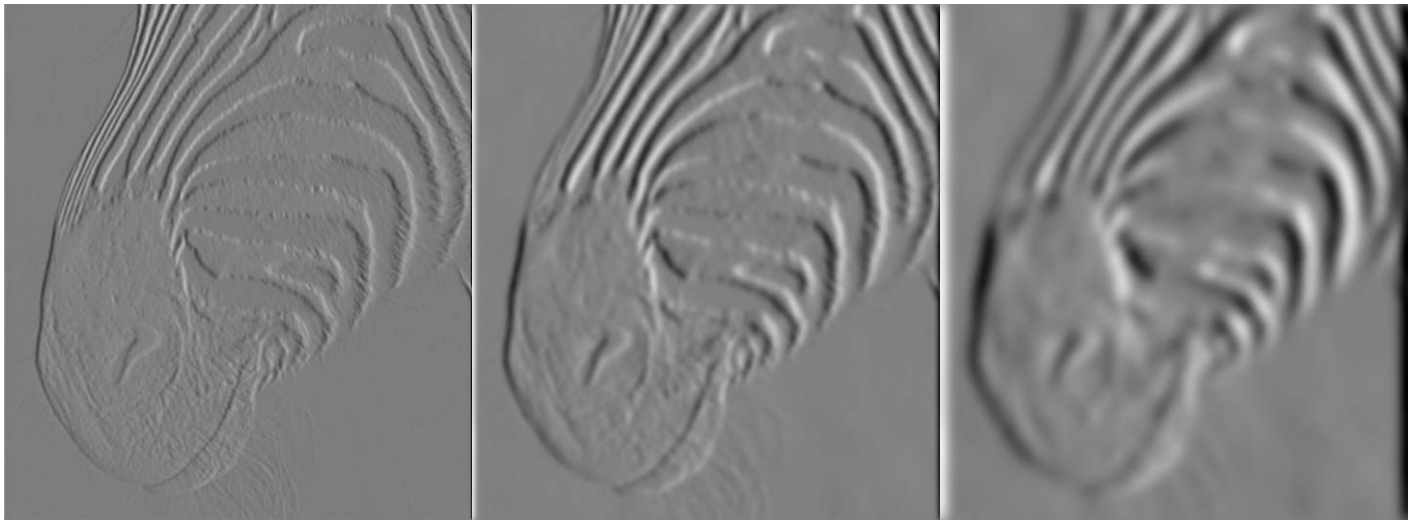
Three steps to perform edge detection:

1. **Noise reduction**, where we try to suppress as much noise as possible, without smoothing away the meaningful edges.

2. **Edge enhancement,** where we apply some kind of filter that responds strongly at edges and weakly elsewhere, so that the edges may be identified as local maxima in the filter's output. One suggestion is to use some kind of high pass filter.

3. **Edge localization**, where we decide which of the local maxima output by the filter are meaningful edges and which are caused by noise

Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Implementation issues

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?

Source: D. Forsyth

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

Stages of Canny Algorithm

- Noise reduction
- Finding the intensity gradient of the image
- Non-maximum suppression
- Tracing edges through the image and hysteresis thresholding

Identifying the local maxima as candidate edge pixels

The difference of the gray values of the two pixels is an estimate of the first derivative of the intensity function (image brightness function) with respect to the spatial variable along the direction of with we take the difference. This is because first derivatives are approximated by first difference in the discrete case.

We also can express gradient calculation as a pair of convolution

$$g_x(x,y) = h_x * f(x,y) \qquad g_y = h_y * f(x,y)$$

In the first output, produced by convolution with mask *hx*, any pixel that has an absolute value larger than values of its left and right neighbors is a candidate pixel to be a vertical edge pixel. In the second output , produced by the convolution with mask *hy*, any pixel that has an absolute value larger than the values of its top and bottom neighbors is a candidate pixel to be a horizontal edge pixel.

The process of identifying the local maxima as candidate edge pixels is called ***non-maxima suppression***.

$\mathbf{h_x} =$

-1	+1

$\mathbf{h_y} =$

-1
+1

Gradient edge detection

Gradient edge detection is the second and more widely used technique. Here, the image is convolved with only two kernels, one estimating the gradient in the *x*-direction, *gx*, the other the gradient in the *y*-direction, *gy*.

A changes of the image function are based on the estimation of **gray level gradient** at a pixel.

The gradient is the two-dimensional equivalent of the first derivative and is defined as the gradient vector.

$$\nabla f(x, y) = g = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

The magnitude of the gray level gradient

There are two important properties associated with the gradient: gradient **magnitude** and **direction** of the gradient

The outputs of the two convolution are squared, added and square rooted to produce the gradient **magnitude**

$$|\boldsymbol{g}| = \sqrt{\boldsymbol{g}_x^2 + \boldsymbol{g}_y^2}$$

The magnitude of the gradient vector is the length of the hypotenuse of the right triangle having sides g_x an g_y, and this reflects the strength of the edge, or edge response, at any given .

The direction of the gray level gradient

From vector analysis, the **direction** of the gradient

$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

This vector is oriented along the direction of the maximum rate of increase of the gray level function $f(x,y)$.

In the localization step , we must to identify the meaningful edges

Whichever operator is used to compute the gradient, the resulting vector contains information about how strong the edge is at that pixel and what its direction is. The edge magnitude is a real number. Any pixel having a gradient that exceeds a specified *threshold value* is said to be an edge pixel, and others are not.

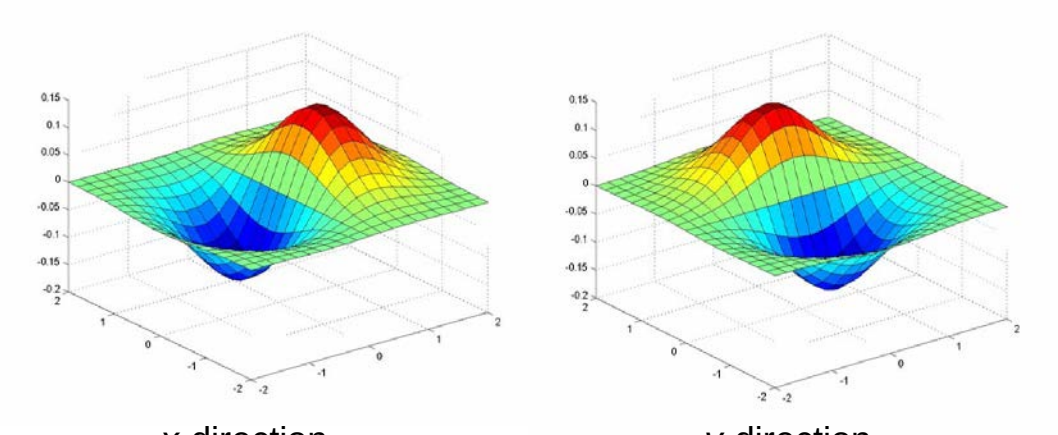
$$g(x,y)_{edge\ \ point} = \begin{cases} 1 \ \ if \ |\nabla f(x,y)| \ > \ \theta \\ 0 \ if \ \ \ \ |\nabla f(x,y)| < \ \theta \end{cases}$$

An alternative technique is to look for local maxima in the gradient image, thus producing one pixel wide edges. A more sophisticated technique is used by the *Canny edge* ***detector***. It first applies a gradient edge detector to the image and then finds the edge pixels using *non-maximal suppression* and *hysteresis tracking*.

Example

original image (Lena)

Derivative of Gaussian filter



x-direction

y-direction

The Canny edge detector is susceptible to noise present in raw unprocessed image data, it uses a filter based on a Gaussian (bell) curve, where the raw image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

Compute Gradients

X-Derivative of Gaussian

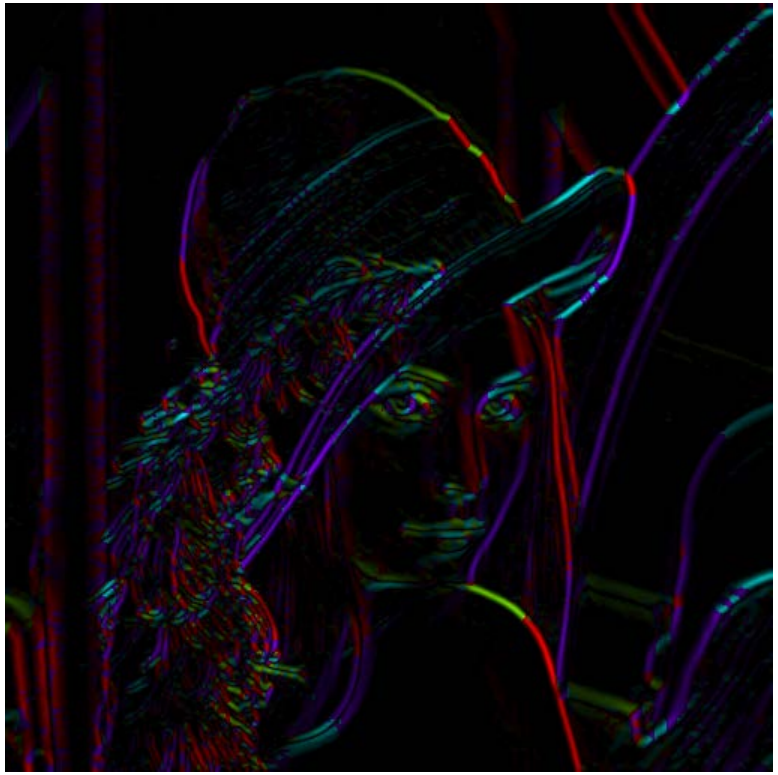
Y-Derivative of Gaussian

Gradient Magnitude

An edge in an image may point in a variety of directions, so the Canny algorithm uses 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.

Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

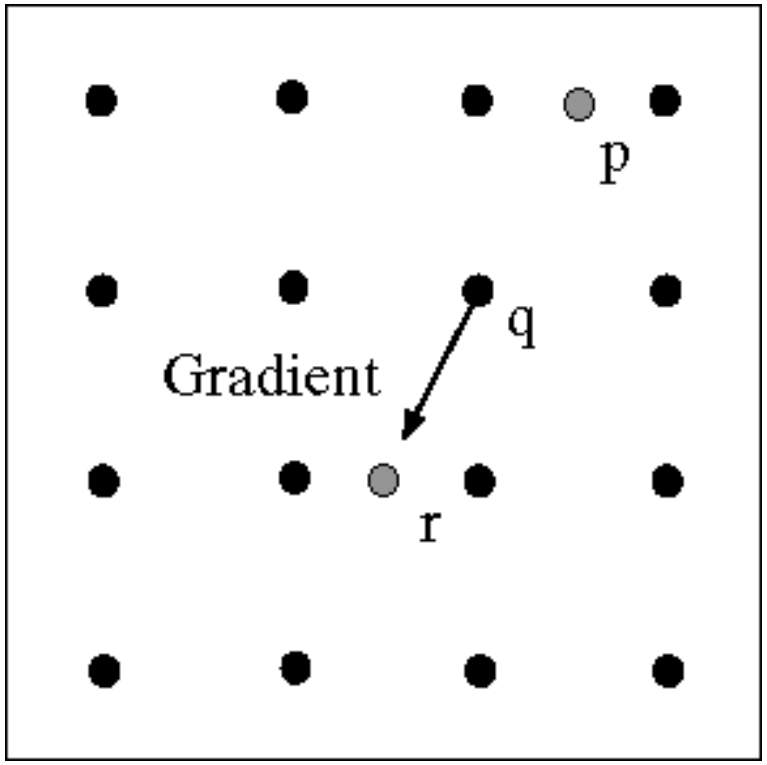


theta = atan2(gy, gx)

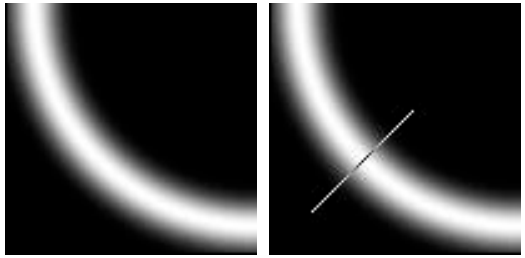
The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

Non-maximum suppression for each orientation

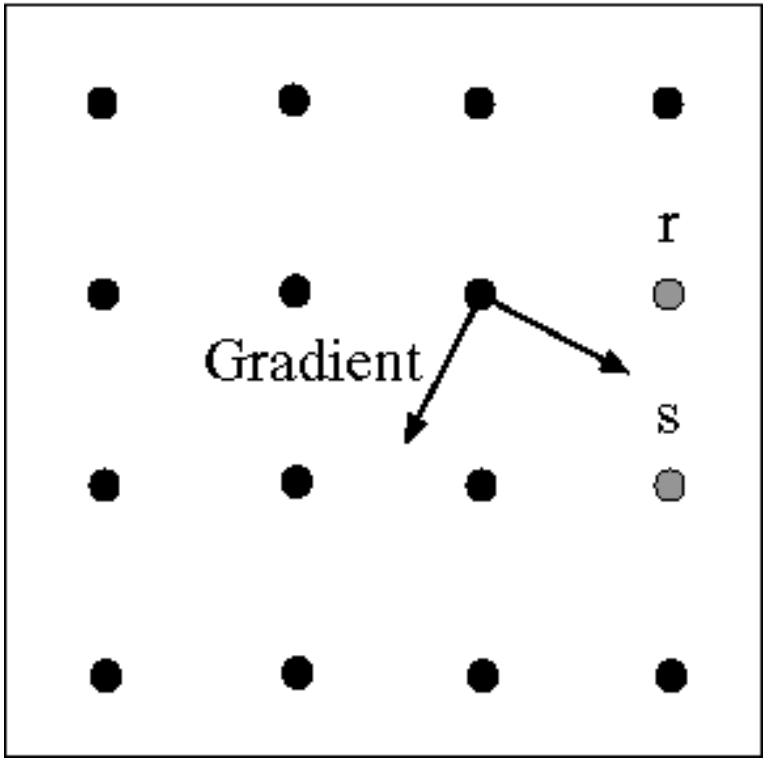
Non-maximum suppression is an edge thinning technique.
Given estimates of the image gradients, a search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction.



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

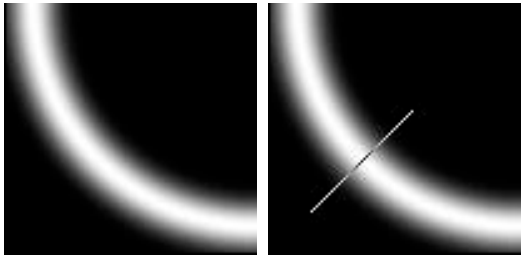


Edge linking



Source: D. Forsyth

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Before Non-max Suppression

After non-max suppression

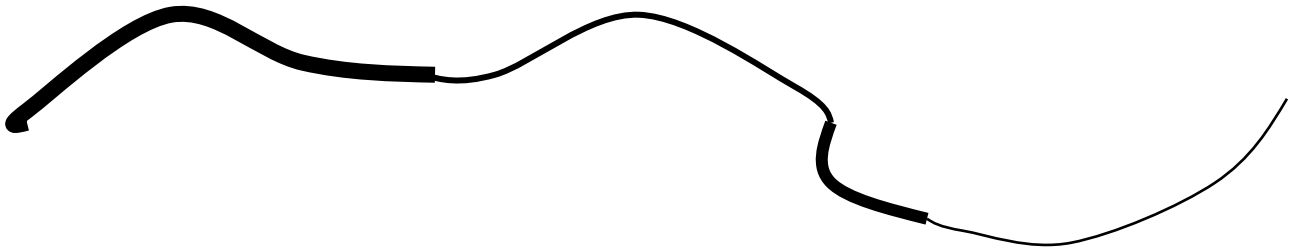
Hysteresis thresholding

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Final Canny Edges

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: edge(image, ‘canny’)

Source: D. Lowe, L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

The most common kernels used for the gradient edge detector are the ***Sobel***, ***Roberts Cross*** and ***Prewitt operators.***

Roberts Operator

- Mark edge point only
- No information about edge orientation
- Work best with binary images
- Primary disadvantage:
  - High sensitivity to noise
  - Few pixels are used to approximate the gradient

Roberts Operator (Cont.)

- First form of Roberts Operator

$$\sqrt{\left[I(r,c)-I(r-1,c-1)\right]^2+\left[I(r,c-1)-I(r-1,c)\right]^2}$$

- Second form of Roberts Operator

$$|I(r,c)-I(r-1,c-1)|+|I(r,c-1)-I(r-1,c)|$$

$$h_1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad h_2=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

Prewitt Operator

- Looks for edges in both horizontal and vertical directions, then combine the information into a single metric.

$$y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Edge Magnitude = $\sqrt{x^2 + y^2}$

Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

Sobel Operator

- Similar to the Prewitt, with different mask coefficients:

$$y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Edge Magnitude = $\sqrt{x^2 + y^2}$ Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

- Sometimes we are interested only in edge magnitudes without regard to their orientations.
- The **Laplacian** may be used.

- The Laplacian has the same properties in all directions and is therefore invariant to rotation in the image.

$$\nabla^2(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \qquad (4.37)$$

- The Laplace operator is a very popular operator approximating the second derivative which gives the gradient magnitude only.

Laplacian Operators

- Edge magnitude is approximated in digital images by a convolution sum.
- The sign of the result (+ or -) from two adjacent pixels provide edge orientation and tells us which side of edge brighter

Second Derivative in 2D: Laplacian (cont'd)

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

$$\nabla^2 f = -4f(i, j) + f(i, j+1) + f(i, j-1) + f(i+1, j) + f(i-1, j)$$

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

Variations of Laplacian

0.5	0.0	0.5
1.0	-4.0	1.0
0.5	0.0	0.5

+

0.5	1.0	0.5
0.0	-4.0	0.0
0.5	1.0	0.5

=

1	1	1
1	-8	1
1	1	1

1	-2	1
1	-2	1
1	-2	1

+

1	1	1
-2	-2	-2
1	1	1

=

2	-1	2
-1	-4	-1
2	-1	2

Properties of Laplacian

- It is an isotropic operator.
- It is cheaper to implement than the gradient (i.e., one mask only).
- It does not provide information about edge direction.
- It is more sensitive to noise (i.e., differentiates twice).