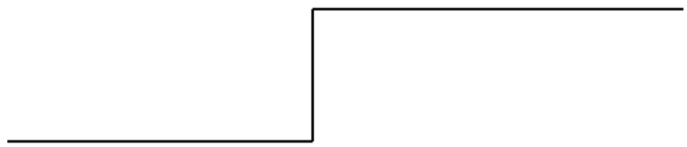
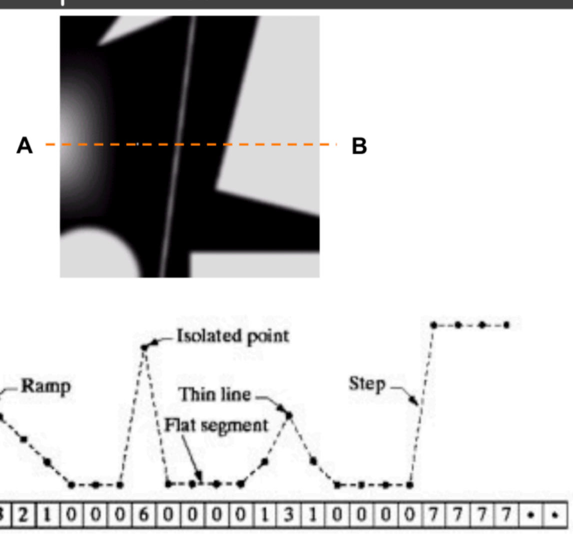
Edge is *Where Change Occurs*

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero.



Spatial Differentiation



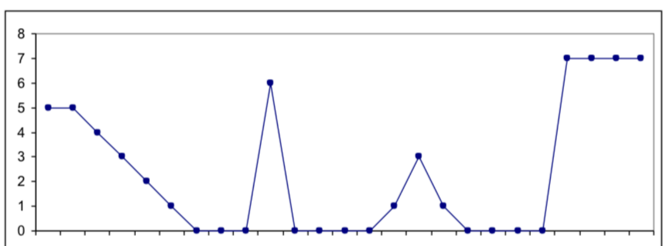
1^{st} Derivative

The formula for the 1^{st} derivative of a function is as follows:

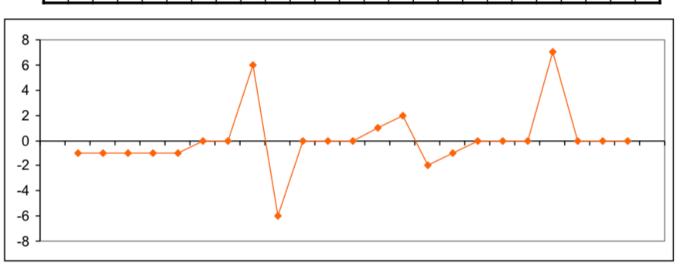
It's just the difference between subsequent values and measures the rate of change of the function

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

1^{st} Derivative (cont…)



5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7
	0	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0	



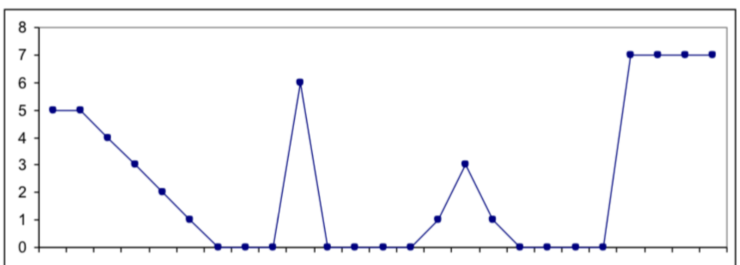
2^{nd} Derivative

The formula for the 2^{nd} derivative of a function is as follows:

Simply takes into account the values both before and after the current value

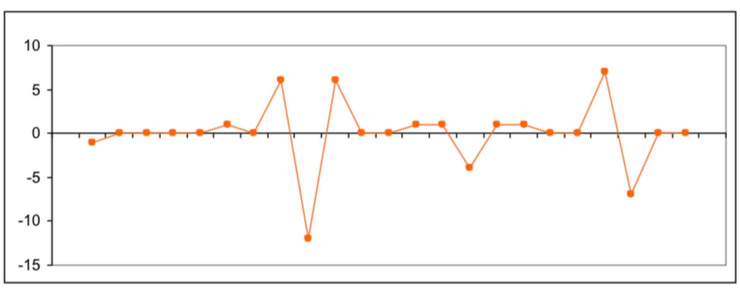
$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

2^{nd} Derivative (cont…)



5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7

	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	



The discrete gradient

- How can we differentiate a *digital* image f[x,y]?
  - Option 1: reconstruct a continuous image, then take gradient
  - Option 2: take discrete derivative (finite difference)

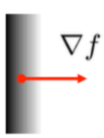
$$\frac{\partial f}{\partial x}[x,y] \approx f[x+1,y] - f[x,y]$$

How can we implement this as a "cross-correlation"?

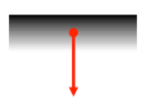
$$H$$

Image gradient

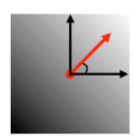
- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

  - how does this relate to the direction of the edge?
- The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

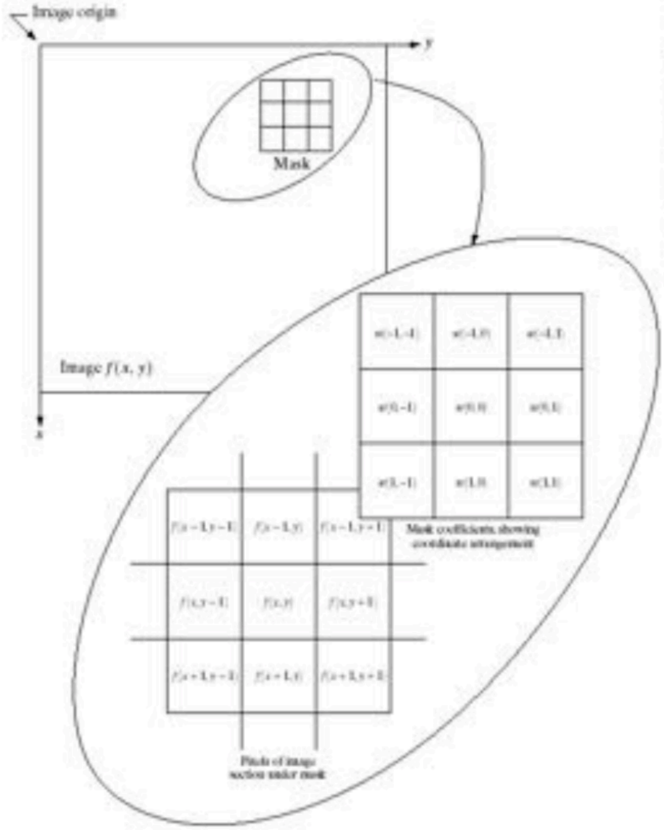
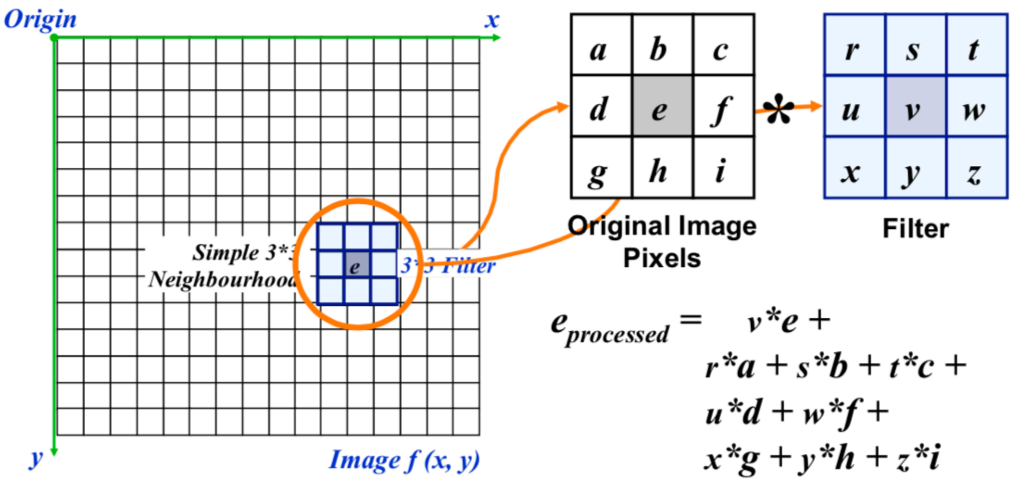


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

FIGURE 3.33 Another representation of a general 3×3 spatial filter mask.

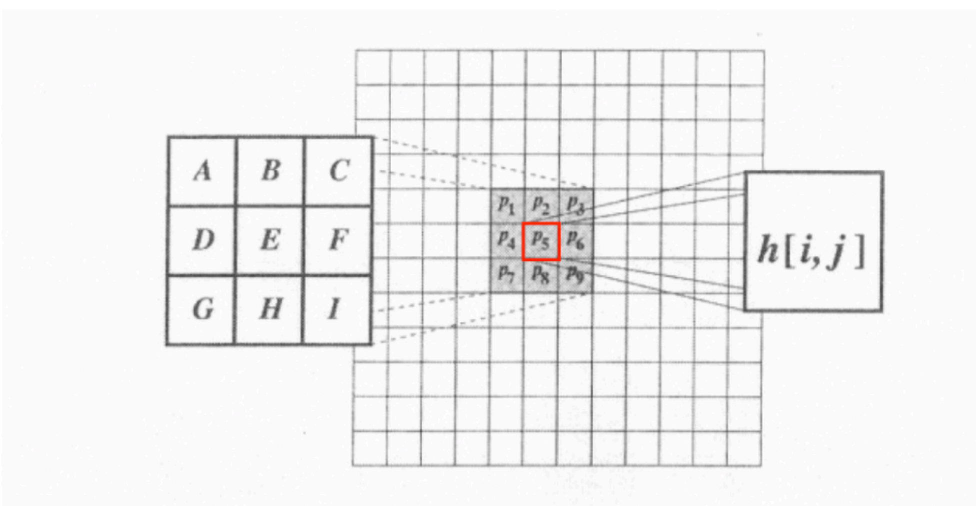
Spatial Filtering Refresher



$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the "processed" image

Example of 3x3 convolution mask



$$h[i,j] = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

2-D Convolution Operation

I

10	11	9	0	0	1
9	$\boxed{11}$	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	$\boxed{10}$				X
X					X
X					X
X					X
X	X	X	X	X	X

H

1/9

1	1	1
1	1	1
1	1	1

1/9.(10x1 + 11x1 + 9x1 + 9x1 + 11x1 + 11x1 + 10x1 + 9x1 + 10x1) =
1/9.(90) = 10

1st Derivative Filtering

For a function *f(x, y)* the gradient of *f* at coordinates *(x, y)* is given as the column vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The Sobel "*Edge operator*"

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

  - The standard defn. of the Sobel operator omits the 1/8 term
    - doesn't make a difference for edge detection
    - the 1/8 term is needed to get the right gradient value, however

Sobel Edge Detector

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial gradient that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input greyscale image.

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gx Gy

P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9

$$|G| = |(P_1 + 2\times P_2 + P_3) - (P_7 + 2\times P_8 + P_9)| + |(P_3 + 2\times P_6 + P_9) - (P_1 + 2\times P_4 + P_7)|$$

From:

http://www.cee.hw.ac.uk/hipr/html

Sobel Example

Sobel filters are typically used for edge detection

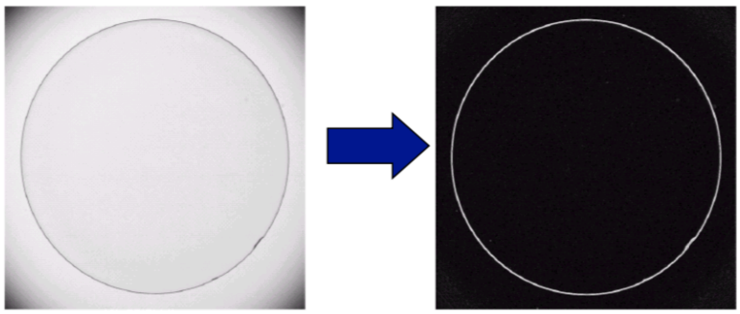
-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

An image of a contact lens which is enhanced in order to make defects (at four and five o'clock in the image) more obvious



ages taken from Gonzalez & Woods, Digital Image Processing (2002)

The Sobel "*Edge operator*"

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

s_y

(a) (b) (c)

Fig. 2. Calculation of edge directional data. (a) Original input image; (b) Sobel edge magnitude; (c) edge directions. Similar brightness value denotes similar edge direction.

[Kang 2003]

Gradient operators

(a)

$$\Delta_1 = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} \qquad \Delta_2 = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

(b)

$$\Delta_1 = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 = \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$$

(c)

$$\Delta_1 = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

(d)

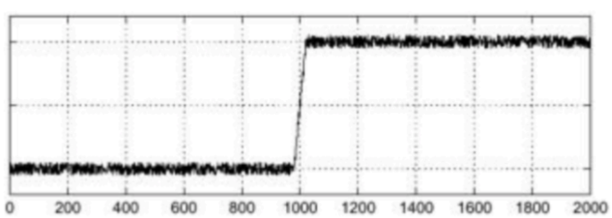
$$\Delta_1 = \begin{matrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{matrix} \qquad \Delta_2 = \begin{matrix} 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -3 & -3 & -3 & -3 \end{matrix}$$

(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

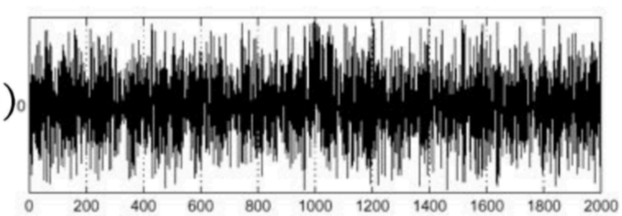
Effects of noise

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

$f(x)$

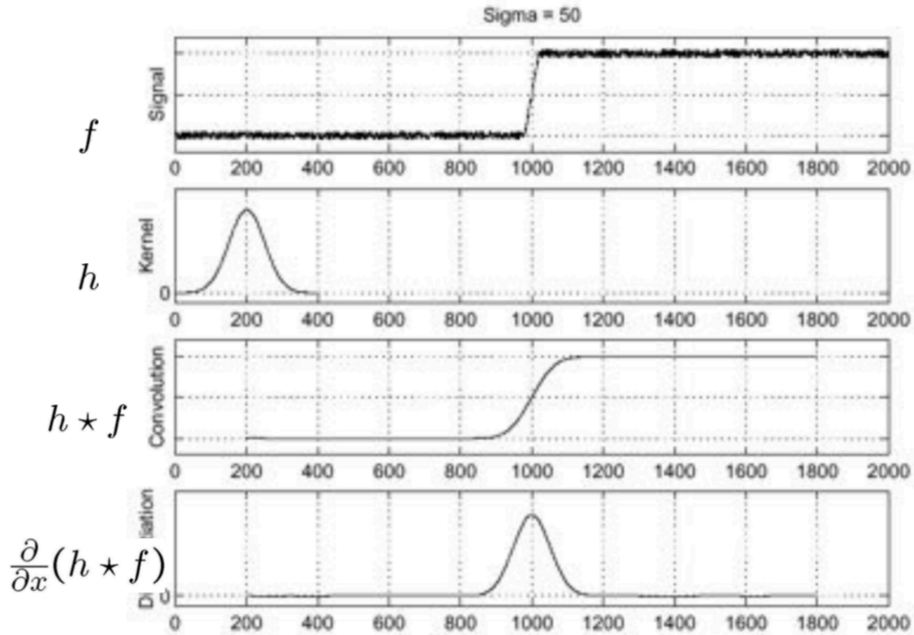


$\frac{d}{dx}f(x)$



- *Where is the edge?*

Solution

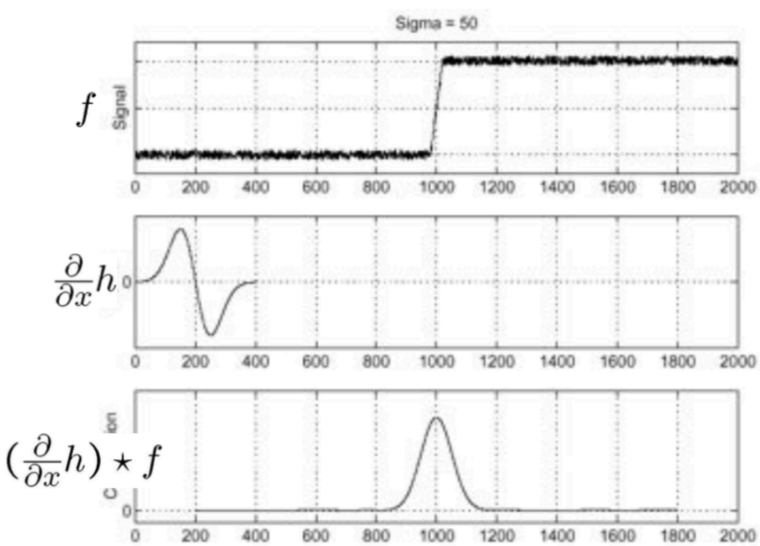


- Where is the edge? -->Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

- **This saves us one operation:**

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$



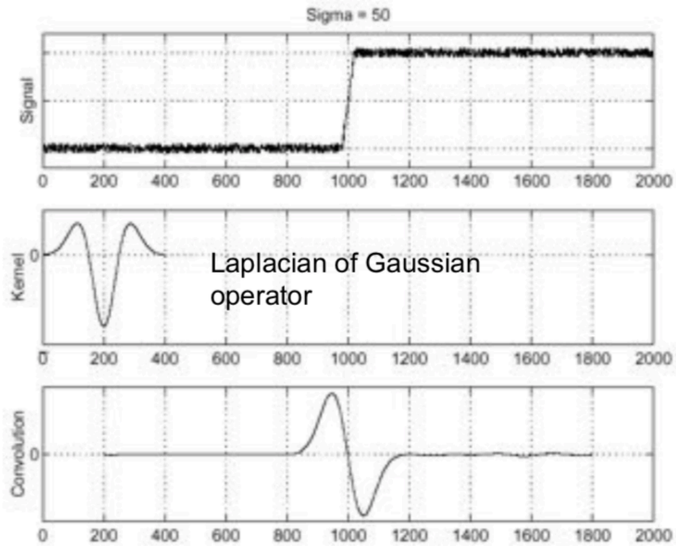
Laplacian of Gaussian

LoG Operator: $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$

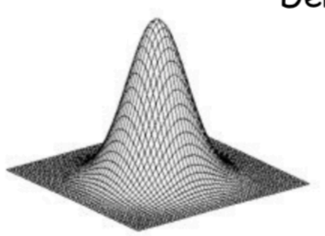


- **Where is the edge?**
- **Zero-crossings of bottom graph**

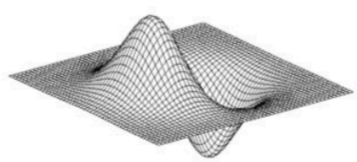
2D edge detection filters

Derivative of Gaussian

Laplacian of Gaussian



Gaussian



$\frac{\partial}{\partial x}h_\sigma(u,v)$

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$\nabla^2 h_\sigma(u,v)$

- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

"Optimal" Edge Detection: Canny

- Assume:
  - Linear filtering
  - Additive Gaussian noise
- Edge detector should have:
  - Good Detection: Filter responds to edge, not noise.
  - Good Localization: detected edge near true edge.
  - Single Response: one per edge.

Optimal Edge Detection: Canny (continued)

- Optimal Detector is similar to *Derivative of Gaussian ("DoG").*
- Detection/Localization trade-off
  - More smoothing improves detection
  - And hurts localization.
- This is what you might guess from

  (detect change) + (remove noise)

The Canny edge detector

- original image (Lena)

The Canny edge detector

- norm of the gradient

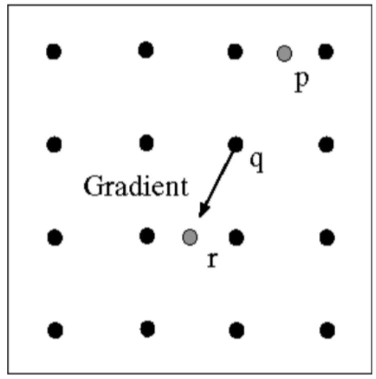
The Canny edge detector

- thresholding

The Canny edge detector

- thinning
- (non-maximum suppression)

Non-maximum suppression

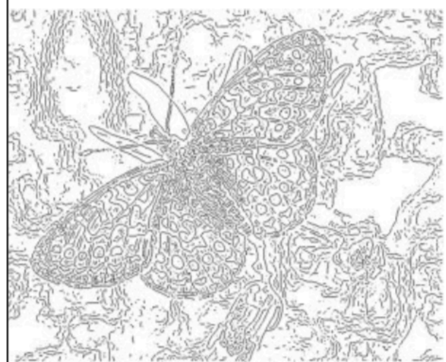


- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r

Effect of Sigma σ (Gaussian kernel size)

original   Canny with $\sigma = 1$ Canny with $\sigma = 2$

- The choice of sigma depends on desired behavior
  - Large sigma ➔ detects large scale edges
  - Small sigma ➔ detects fine features



fine scale
high
threshold