

# Basic Filters (7)

- Convolution/correlation/Linear filtering
- Gaussian filters
- Smoothing and noise reduction
- First derivatives of Gaussian
- Second derivative of Gaussian: Laplacian
- Oriented Gaussian filters
- Steerability



# Convolution

- 1D Formula:

$$(h * f)(x) = \int_u h(x - u)f(u)du$$

$$(h * f)[x] = \sum_i h[x - i]f[i]$$

- 2D Formula:

$$(h * f)(x, y) = \int_u h(x - u, y - v)f(u, v)du$$

$$(h * f)[x, y] = \sum_i h[x - i, y - j]f[i, j]$$

- Example on the web:

- <http://www.jhu.edu/~signals/convolve/>

Original Image



Slight Blurring



Kernel:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

More Blurring



Kernel:

0.0	0.0	0.0	0.0	0.0
4	4	4	4	4
0.0	0.0	0.0	0.0	0.0
4	4	4	4	4
0.0	0.0	0.0	0.0	0.0
4	4	4	4	4
0.0	0.0	0.0	0.0	0.0
4	4	4	4	4
0.0	0.0	0.0	0.0	0.0
4	4	4	4	4

Lots of Blurring



Kernel:  
15 x 15  
matrix of  
value 1/225

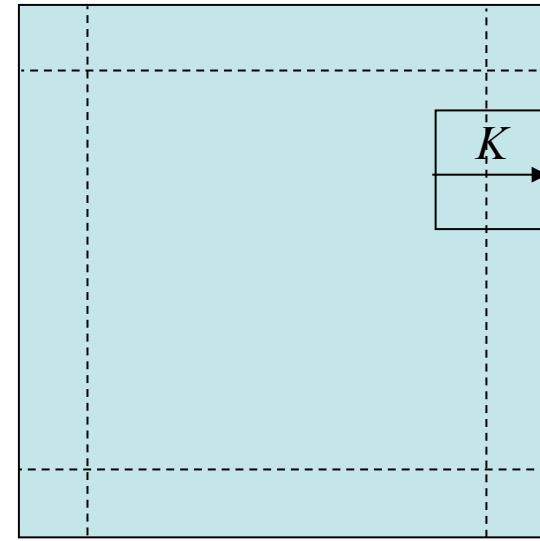
# Basic Properties

- Commutes:  $f * g = g * f$
- Associative:  $(f * g) * h = f * (g * h)$
- Linear:  $(af + bg) * h = a f * h + b g * h$
- Shift invariant:  $f_t * h = (f * h)_t$
- Only operator both linear and shift invariant
- Differentiation:  $\frac{\partial}{\partial x} (f * g) = \frac{\partial f}{\partial x} * g$

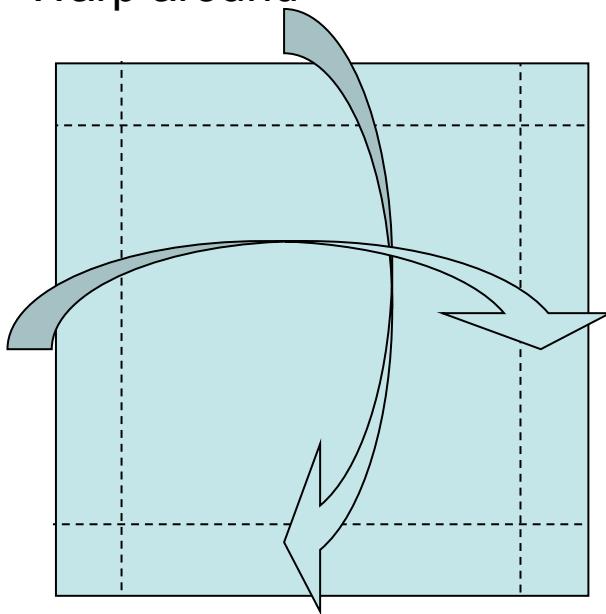
# Practicalities (discrete convolution/correlation)

- MATLAB: `conv` (1D) or `conv2` (2D), `corr`
- Border issues:
  - When applying convolution with a  $K \times K$  kernel, the result is undefined for pixels closer than  $K$  pixels from the border of the image
- Options:

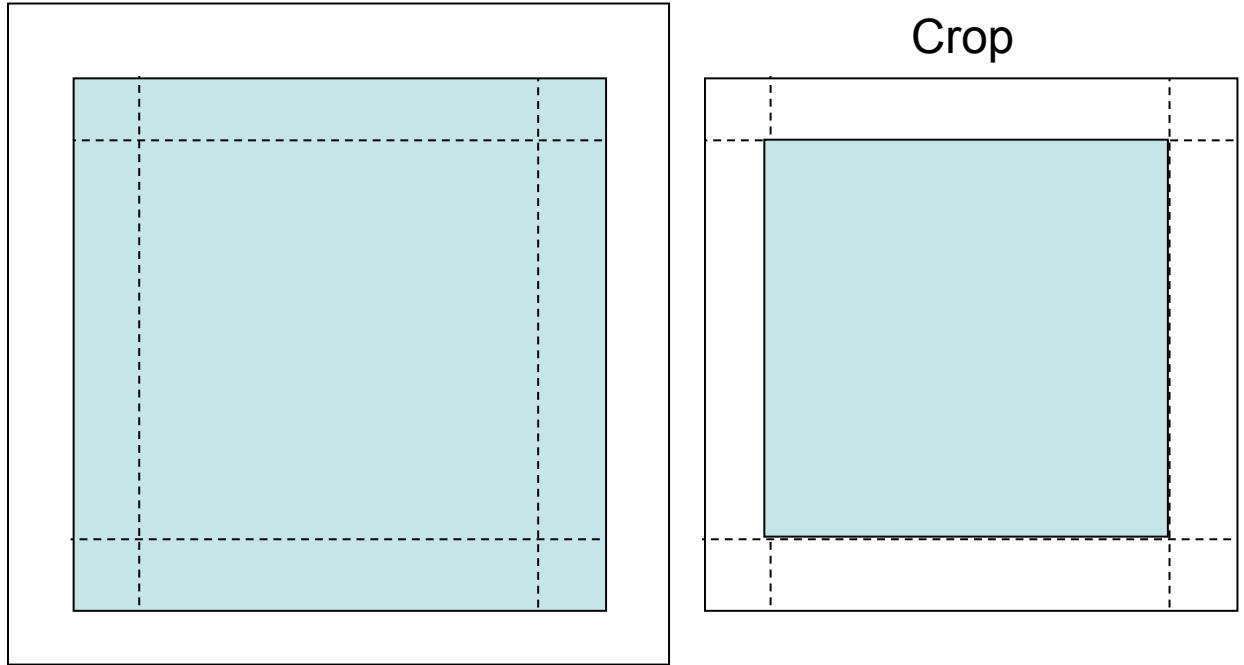
Expand/Pad



Warp around

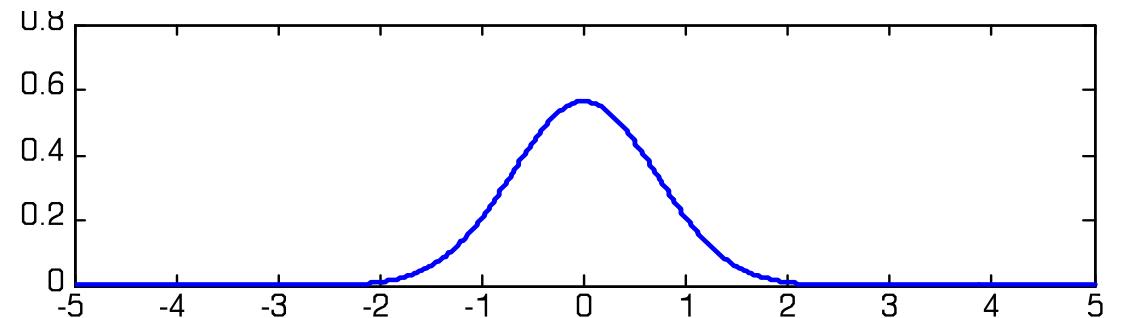


Crop



1-D:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

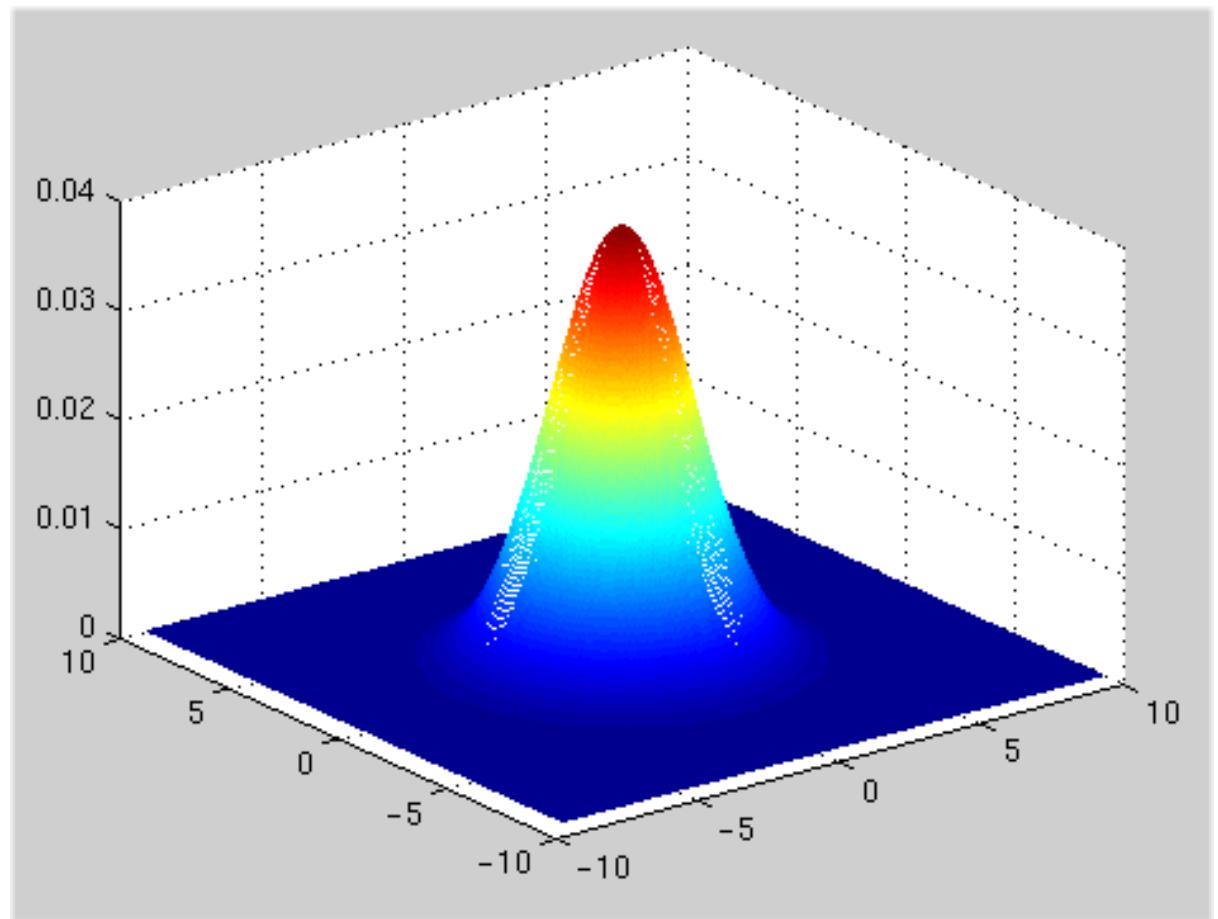


2-D:

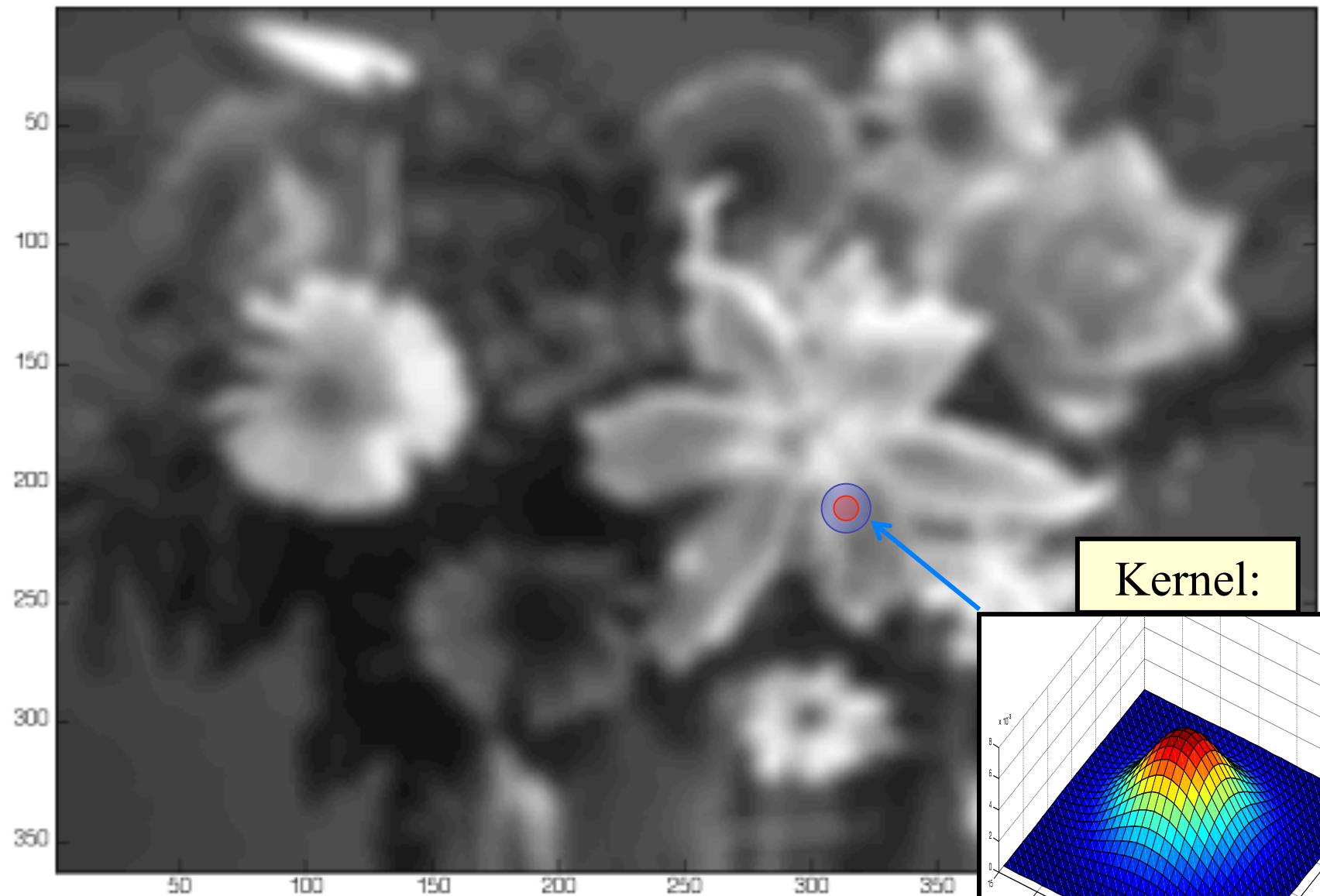
$$G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Slight abuse of notations:  
We ignore the normalization  
constant such that

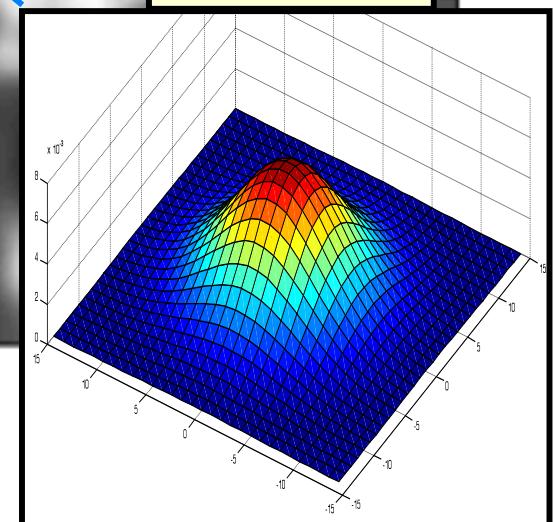
$$\int g(x)dx = 1$$



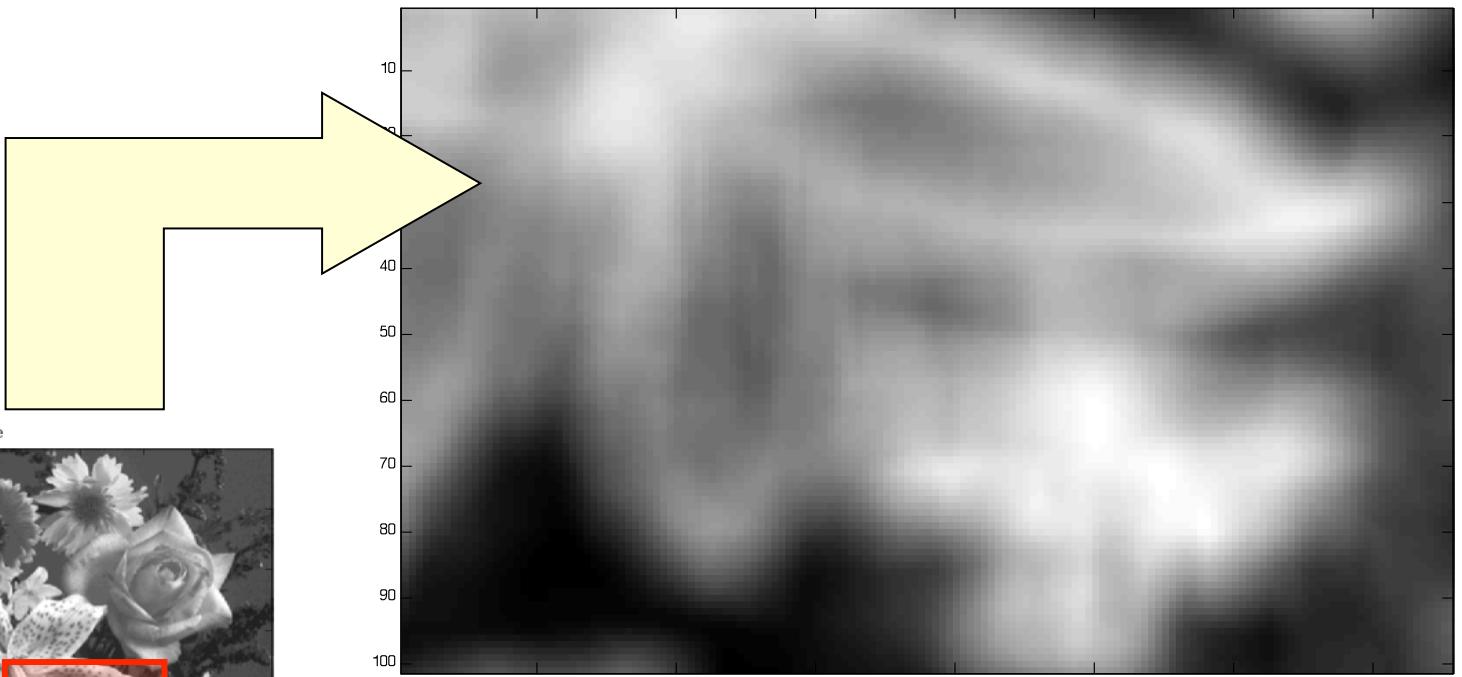
Gaussian Blurring,  $\sigma = 5$



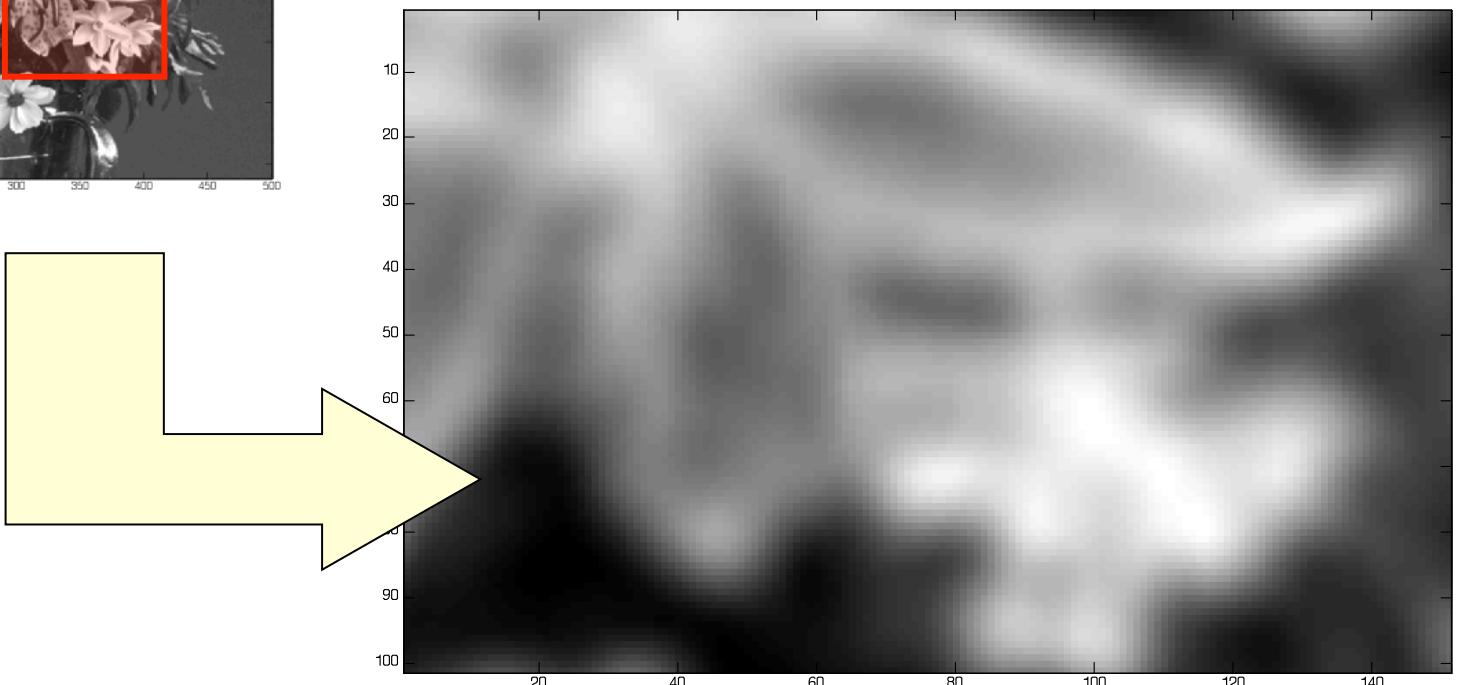
Kernel:



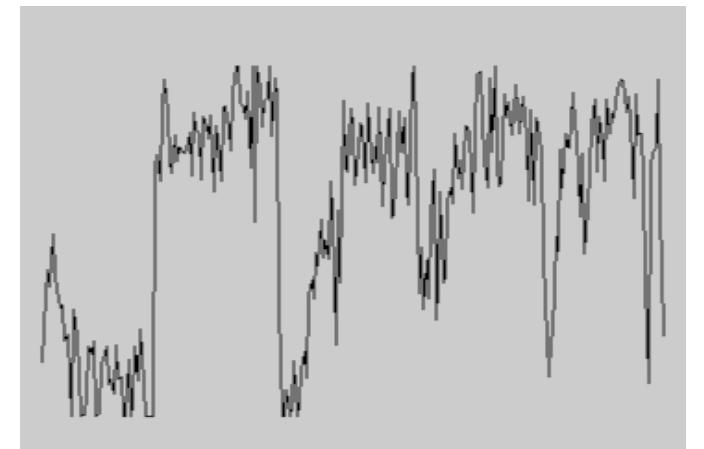
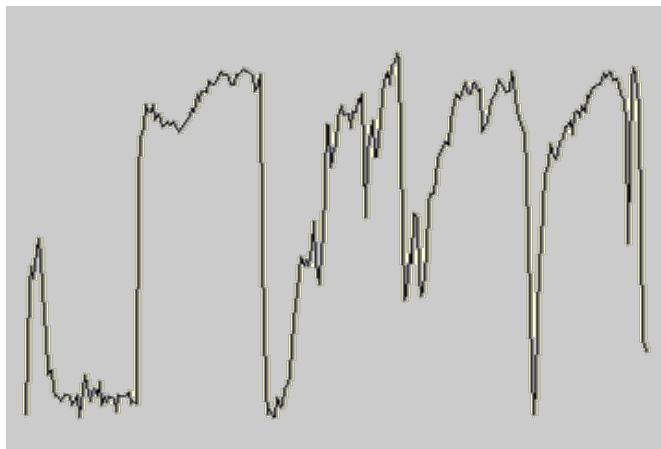
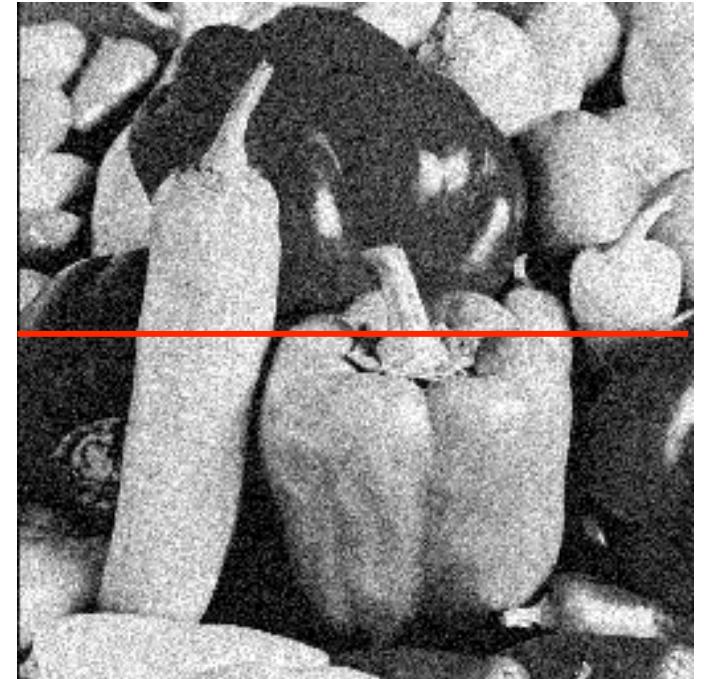
## Simple Averaging



## Gaussian Smoothing



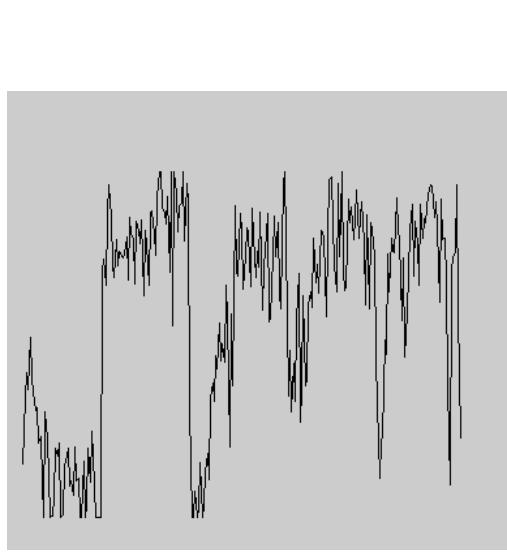
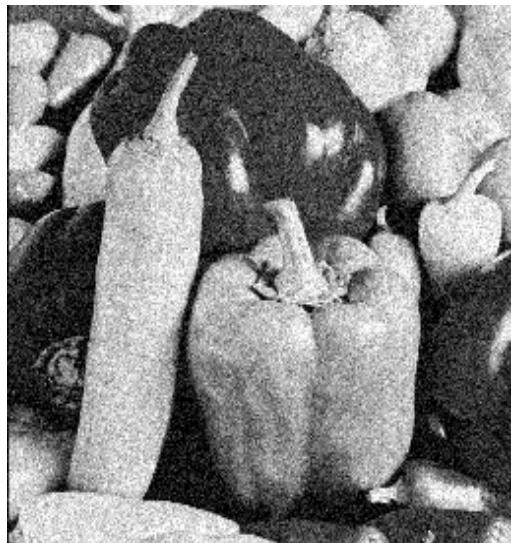
# Image Noise



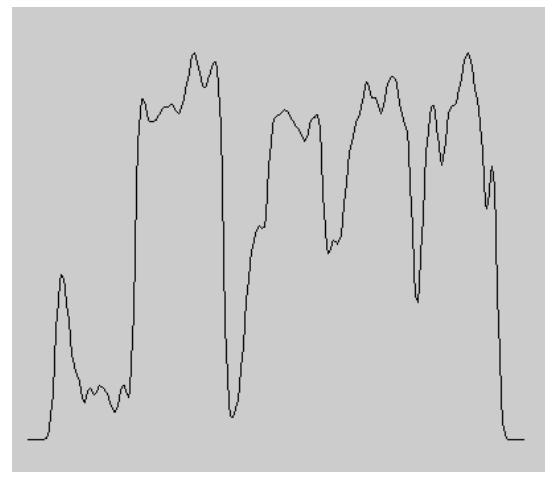
$$f(x, y) = \underbrace{\hat{f}(x, y)}_{\text{Ideal Image}} + \underbrace{\eta(x, y)}_{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:  
 $\eta(x, y) \sim \mathcal{N}(\mu, \sigma)$

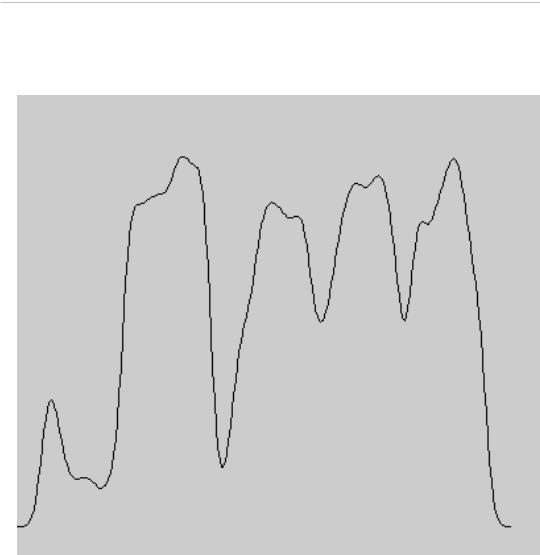
# Gaussian Smoothing to Remove Noise



No smoothing

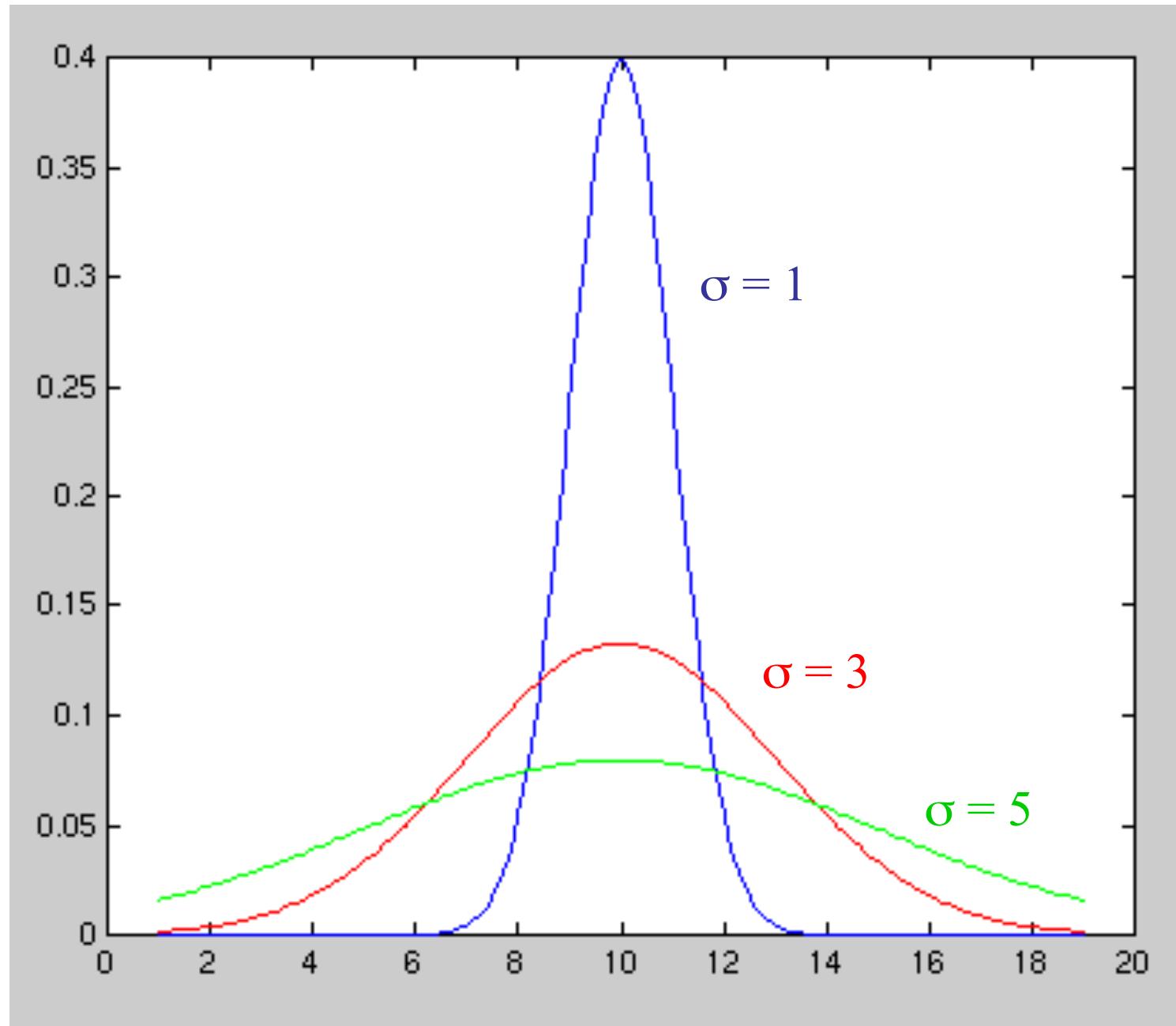


$\sigma = 2$



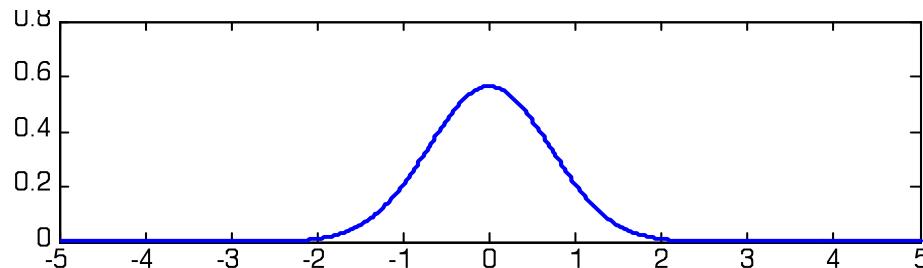
$\sigma = 4$

## Shape of Gaussian filter as function of $\sigma$

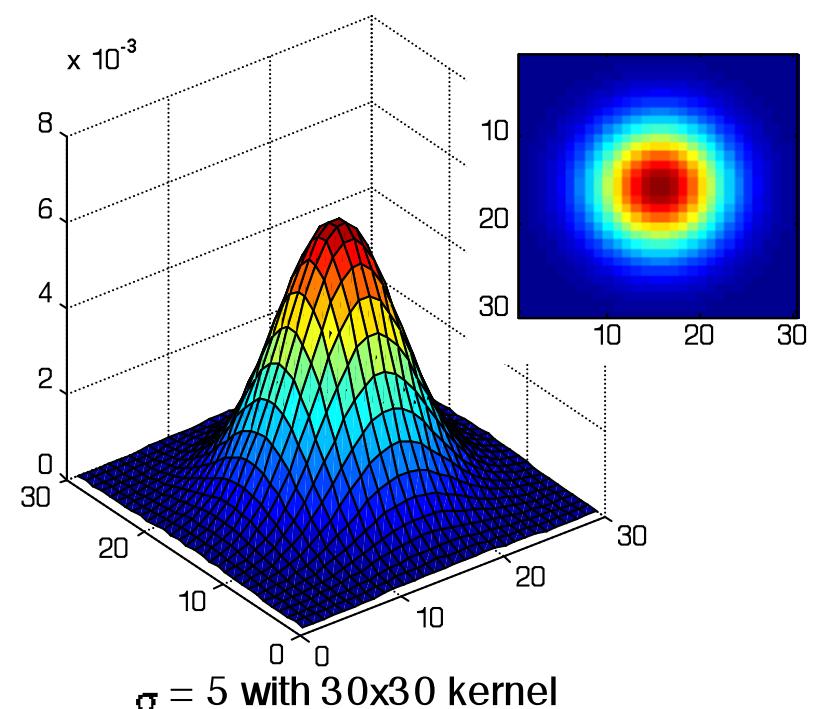
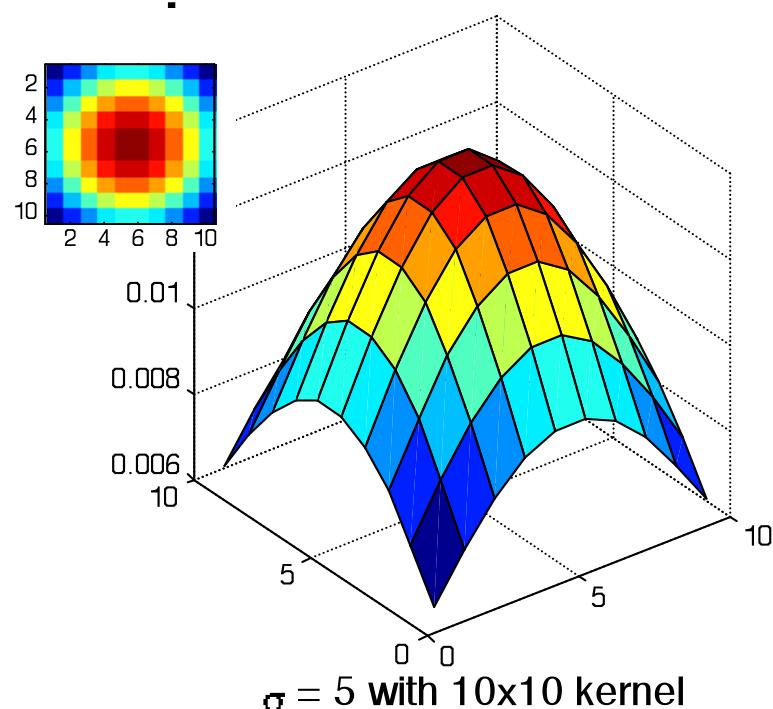


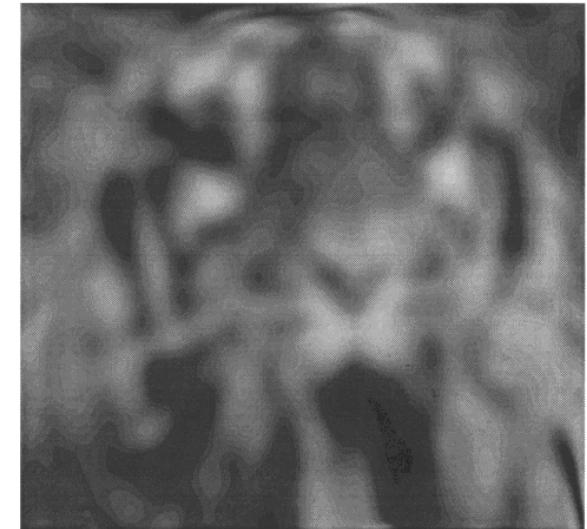
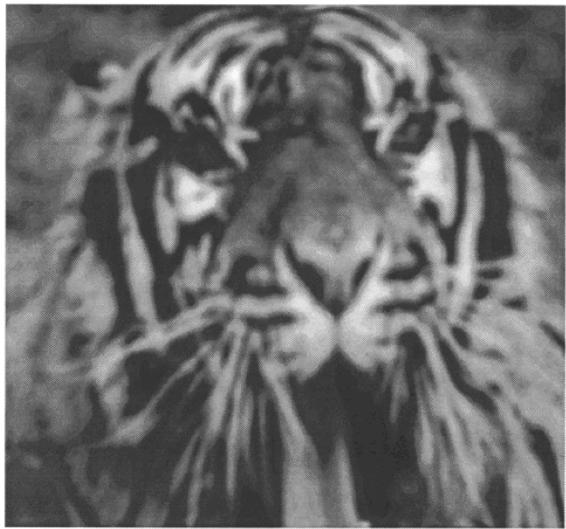
# Note about Finite Kernel Support

- Gaussian function has infinite support



- In discrete filtering, we have finite kernel





---

Increasing  $\sigma$

# Basic Properties

- Gaussian removes “high-frequency” components from the image → “low pass” filter
- Larger  $\sigma$  remove more details
- Combination of 2 Gaussian filters is a Gaussian filter:

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sigma} \quad \sigma^2 = \sigma_1^2 + \sigma_2^2$$

- Separable filter:

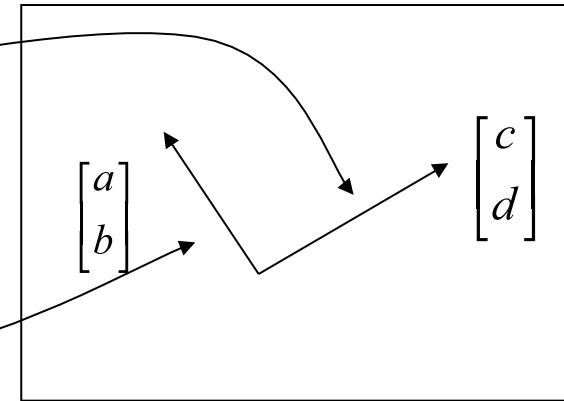
$$G_{\sigma} * f = g_{\sigma \rightarrow} * g_{\sigma \uparrow} * f$$

- Critical implication: Filtering with a  $N \times N$  Gaussian kernel can be implemented as two convolutions of size  $N \rightarrow$  reduction quadratic to linear → *must* be implemented that way

# Oriented Gaussian Filters

- $G_\sigma$  smoothes the image by the same amount in all directions
- If we have some information about preferred directions, we might want to smooth with some value  $\sigma_1$  in the direction defined by the unit vector  $[a \ b]$  and by  $\sigma_2$  in the direction defined by  $[c \ d]$

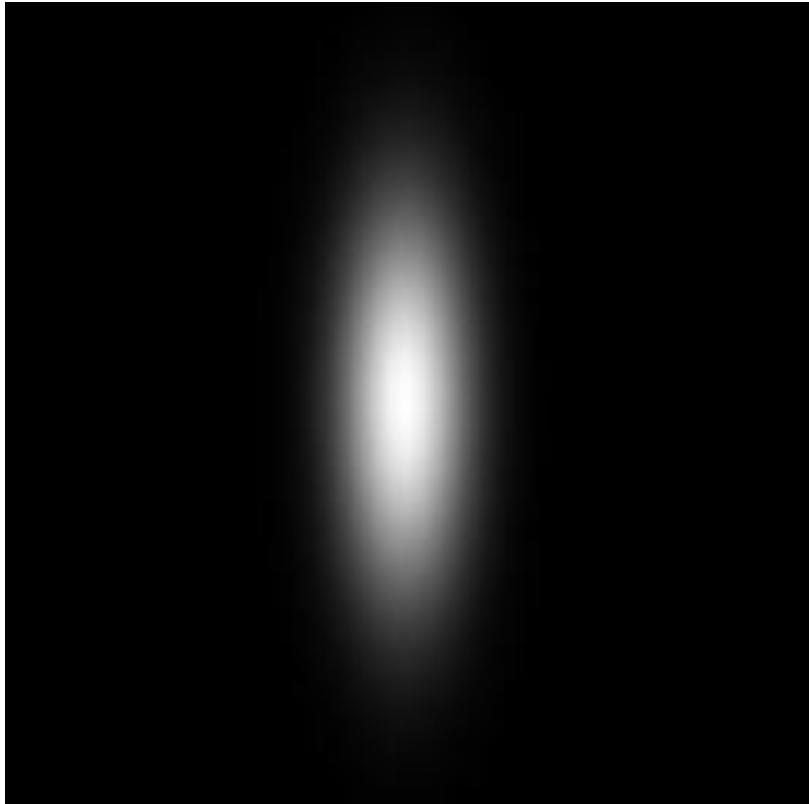
$$G = e^{-\frac{(ax+by)^2}{2\sigma_1^2} - \frac{(cx+dy)^2}{2\sigma_2^2}}$$

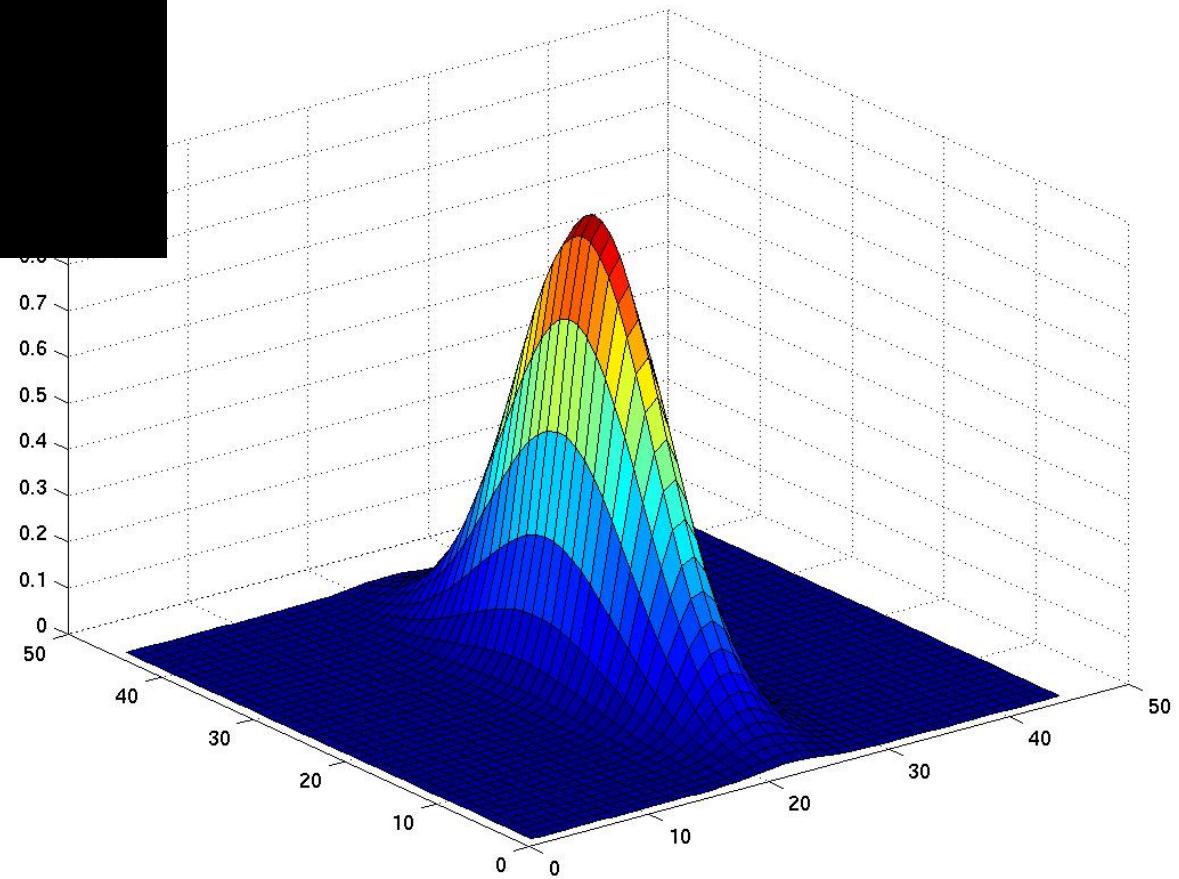


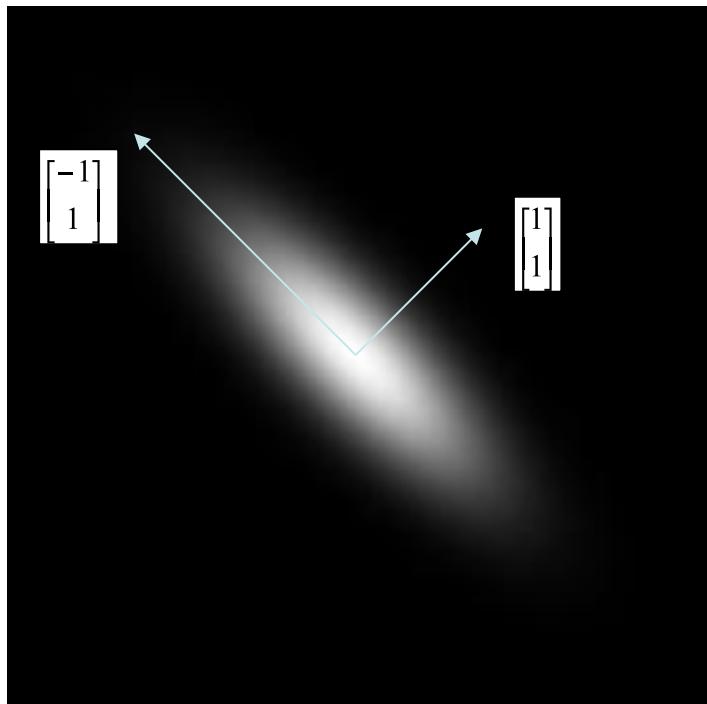
- We can write this in a more compact form by using the standard multivariate Gaussian notation:

$$G_\Sigma = e^{-\frac{X^T \Sigma^{-1} X}{2}} \quad X = \begin{bmatrix} x \\ y \end{bmatrix}$$

- The two (orthogonal) directions of filtering are given by the eigenvectors of  $\Sigma$ , the amount of smoothing is given by the square root of the corresponding eigenvalues of  $\Sigma$ .

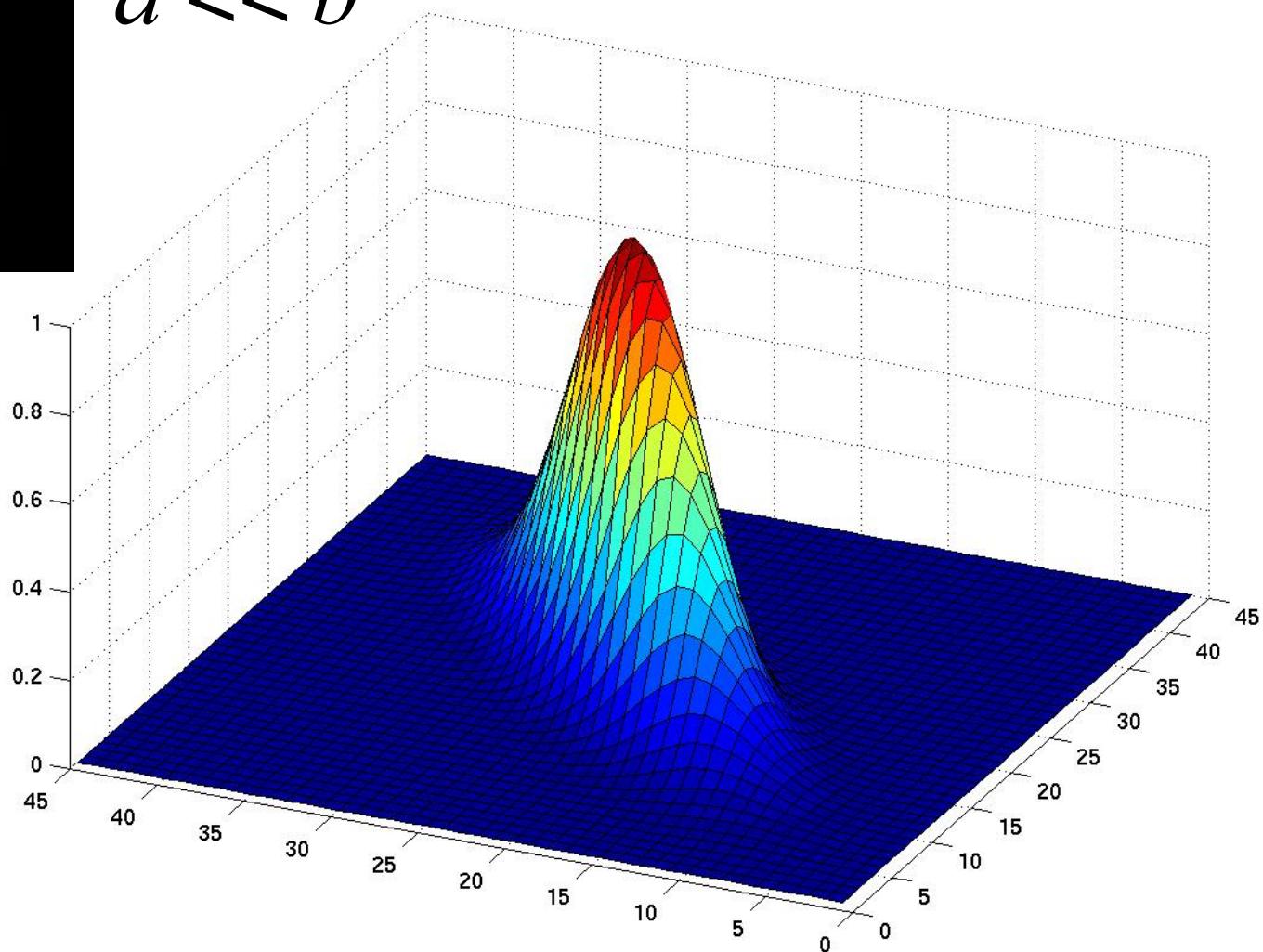

$$\Sigma = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad a \ll b$$

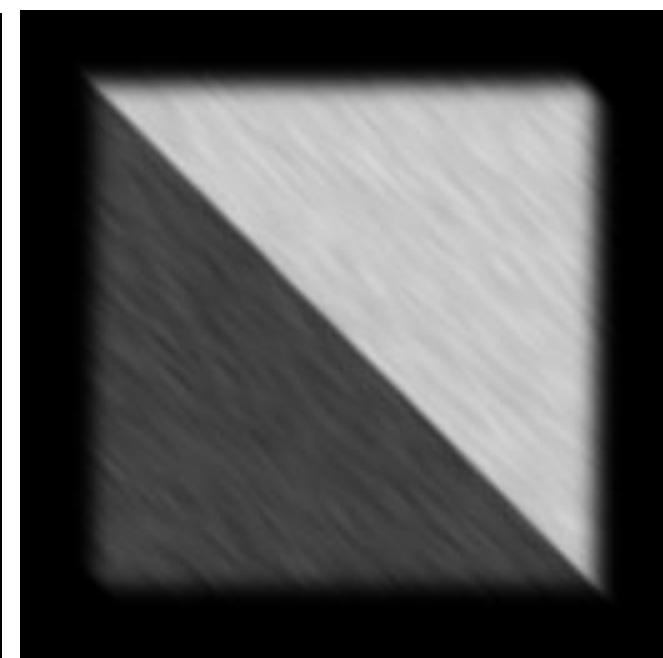
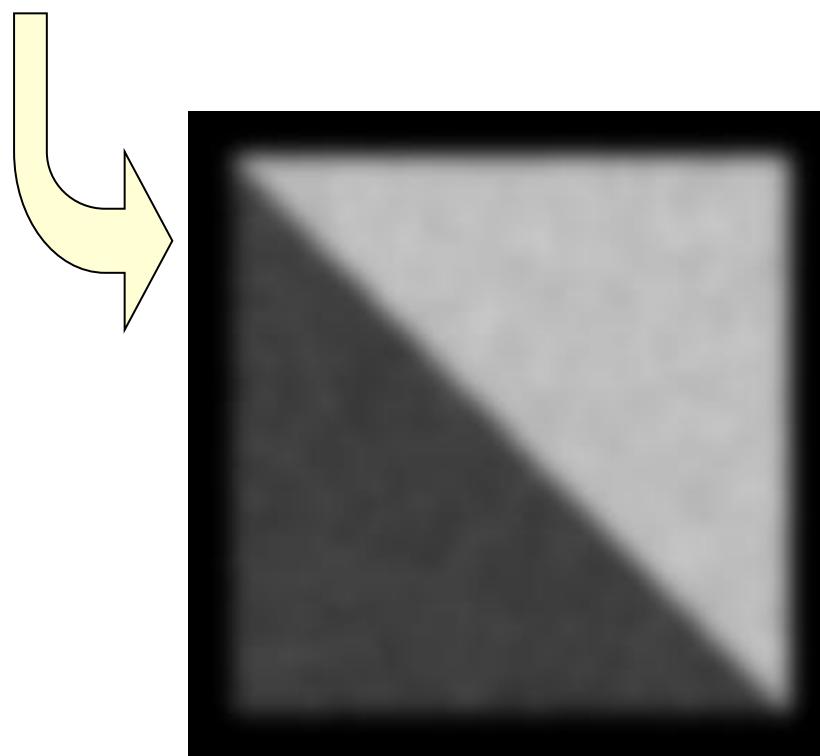
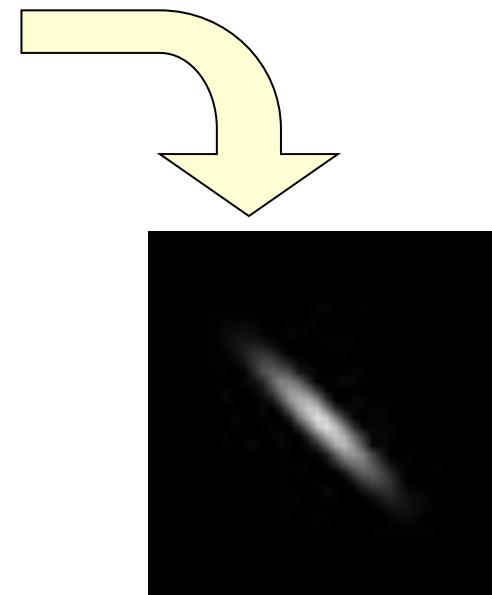
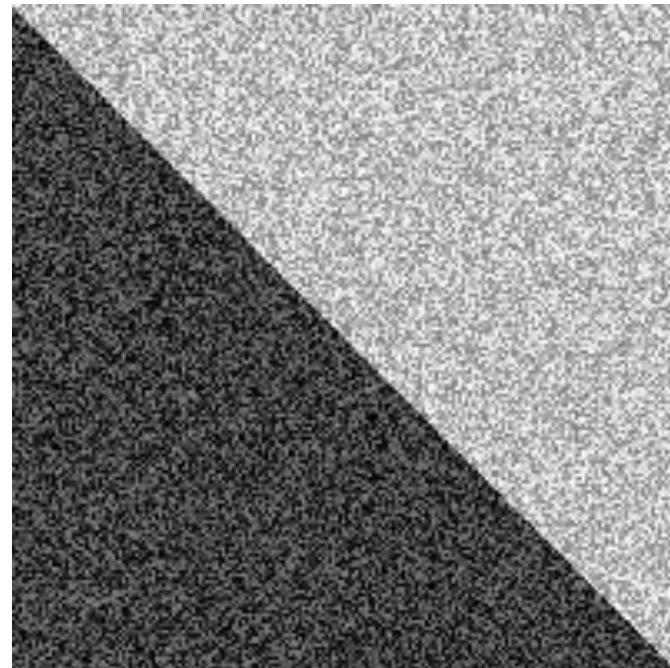
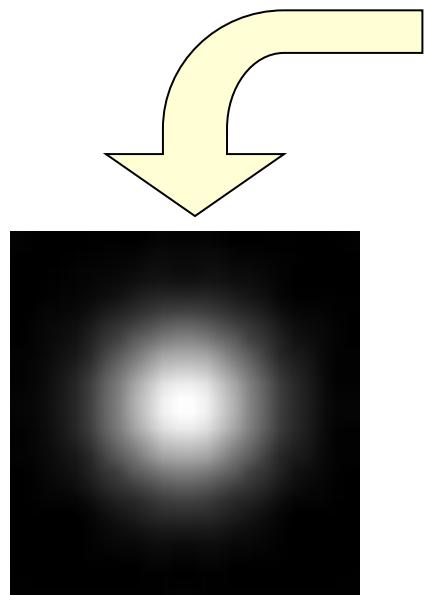




$$\Sigma = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$a \ll b$$





# Image Derivatives

- Image Derivatives
- Derivatives increase noise
- Derivative of Gaussian
- Laplacian of Gaussian (LOG)

# Image Derivatives

- We want to compute, at each pixel  $(x, y)$  the derivatives:
- In the discrete case we could take the difference between the left and right pixels:

$$\frac{\partial I}{\partial x} \approx I(i+1, j) - I(i-1, j)$$

- Convolution of the image by

$$\partial_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline \end{array}$$

- Problem: Increases noise

$$I(i+1, j) - I(i-1, j) = \hat{I}(i+1, j) - \hat{I}(i-1, j) + n_+ + n_-$$

Difference between  
Actual image values

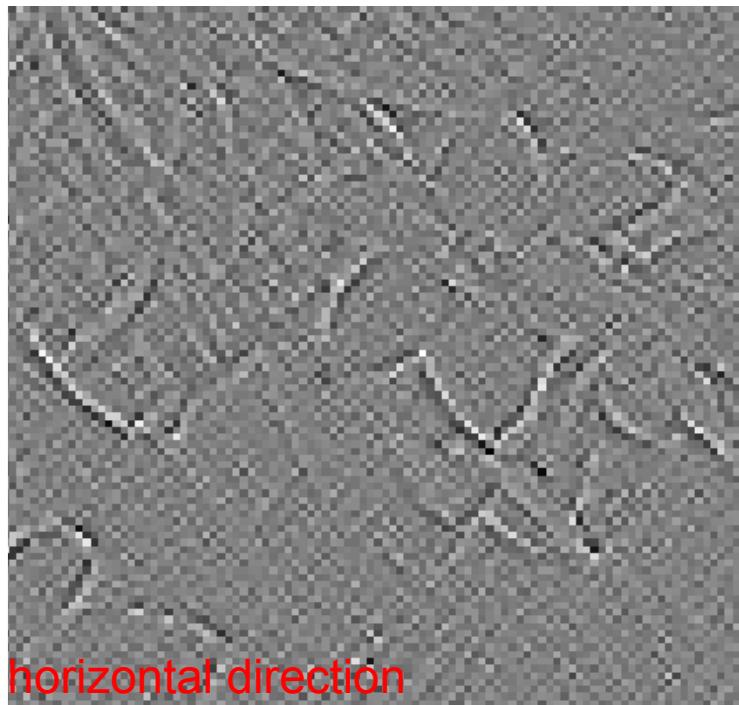
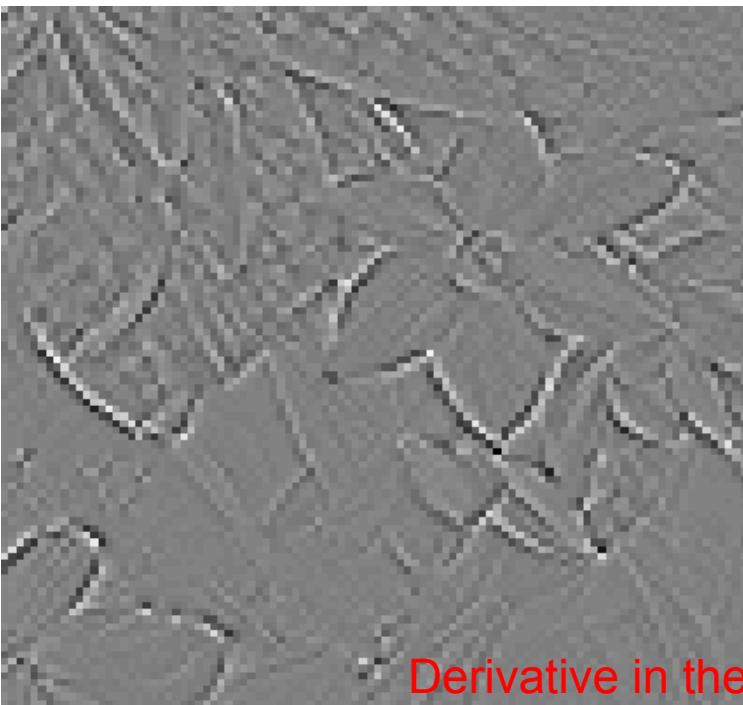
True difference  
(derivative)

Twice the amount of  
noise as in the original  
image

Original Image



Noise Added



Derivative in the horizontal direction

# Smooth Derivatives

- Solution: First smooth the image by a Gaussian  $G_\sigma$  and then take derivatives:

$$\frac{\partial f}{\partial x} \approx \frac{\partial(G_\sigma * f)}{\partial x}$$

- Applying the differentiation property of the convolution:

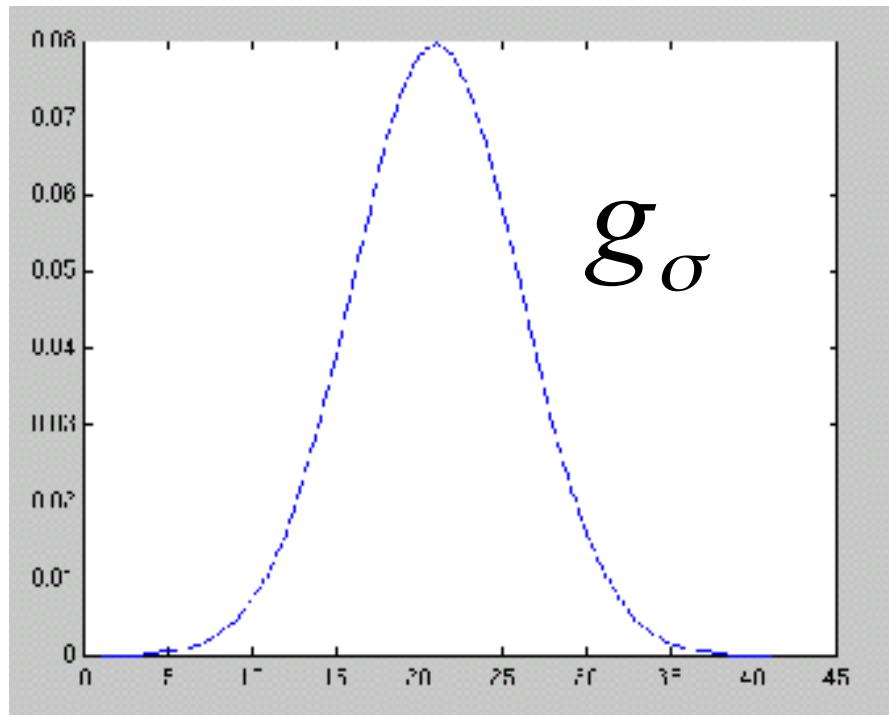
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

- Therefore, taking the derivative in  $x$  of the image can be done by convolution with the derivative of a Gaussian:

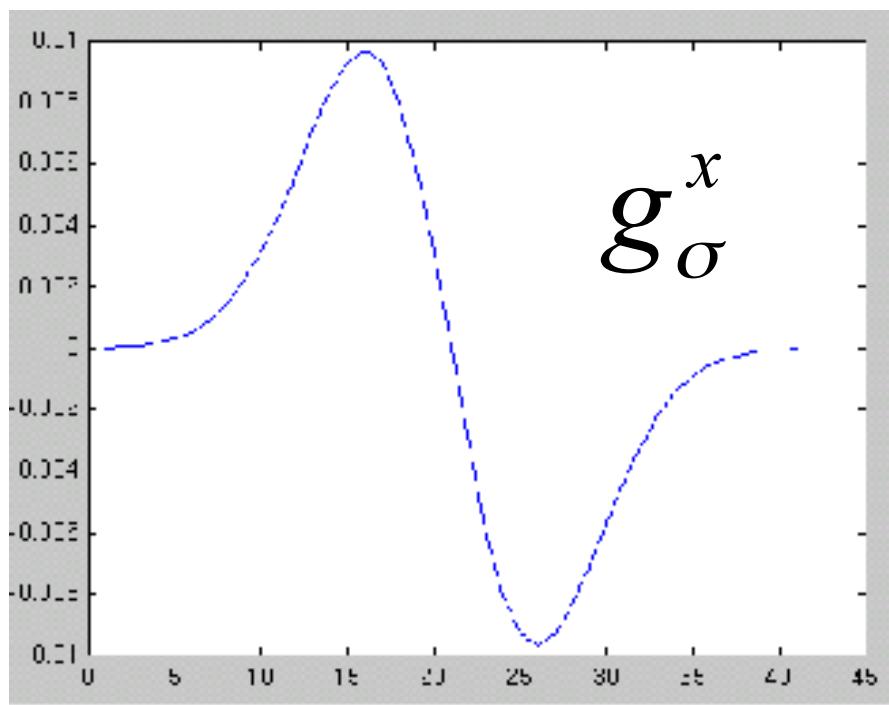
$$G_\sigma^x = \frac{\partial G_\sigma}{\partial x} = xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Crucial property: The Gaussian derivative is also separable:

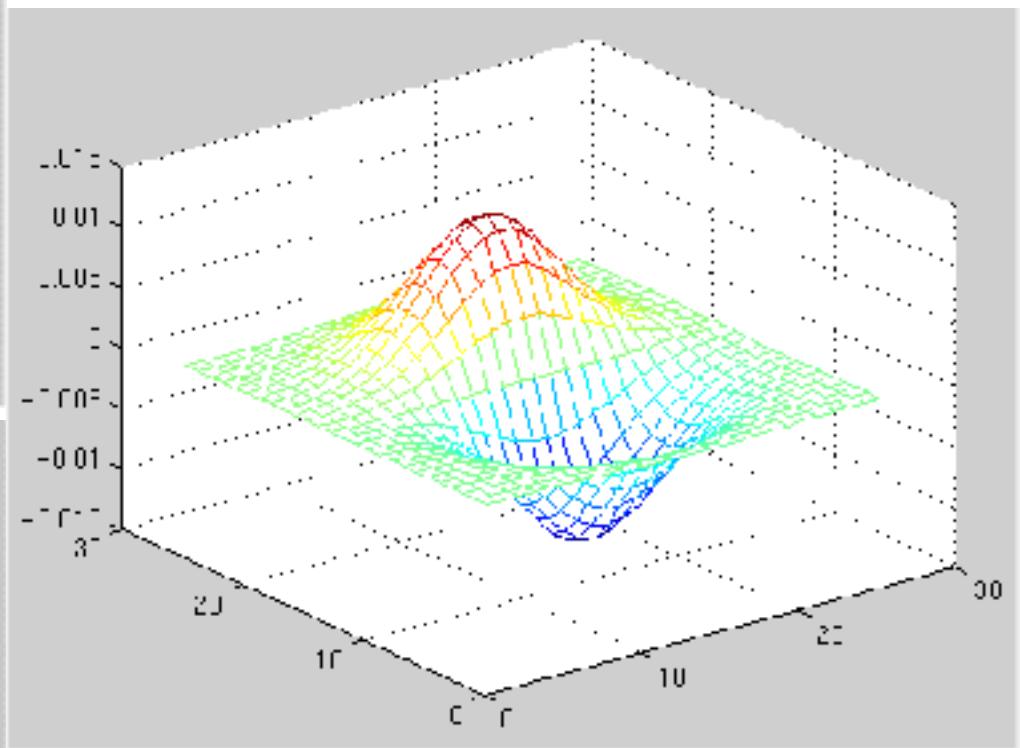
$$G_\sigma^x * f = g_\sigma^x * g_{\sigma \uparrow} * f$$



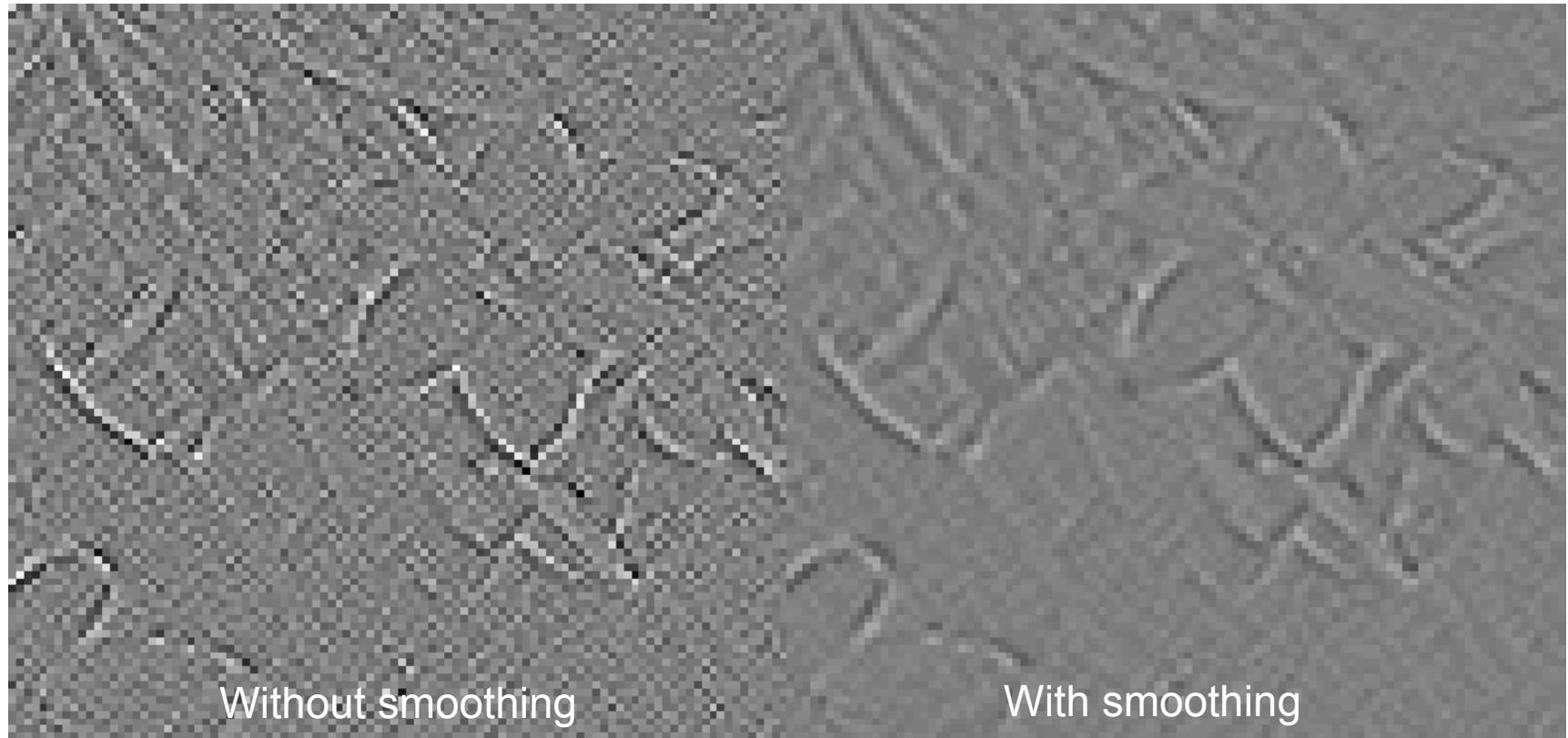
$g_\sigma$



$g_\sigma^x$



# Derivative + Smoothing



Better but still blurs away edge information

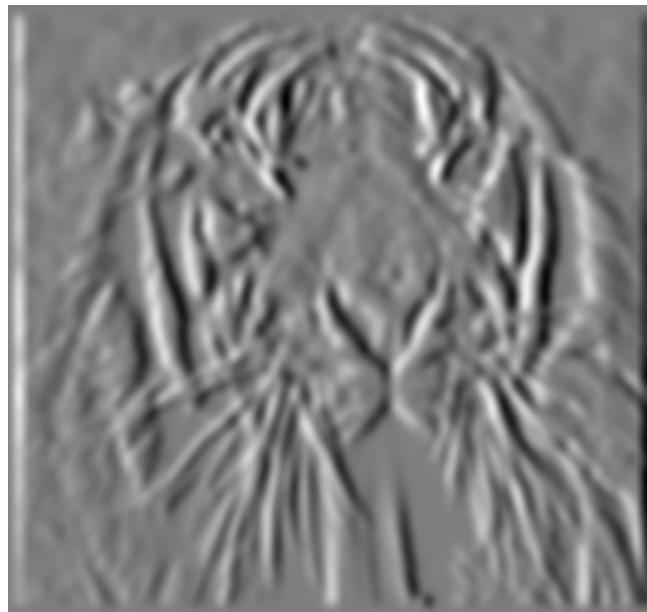
## Applying the first derivative of Gaussian

$I$

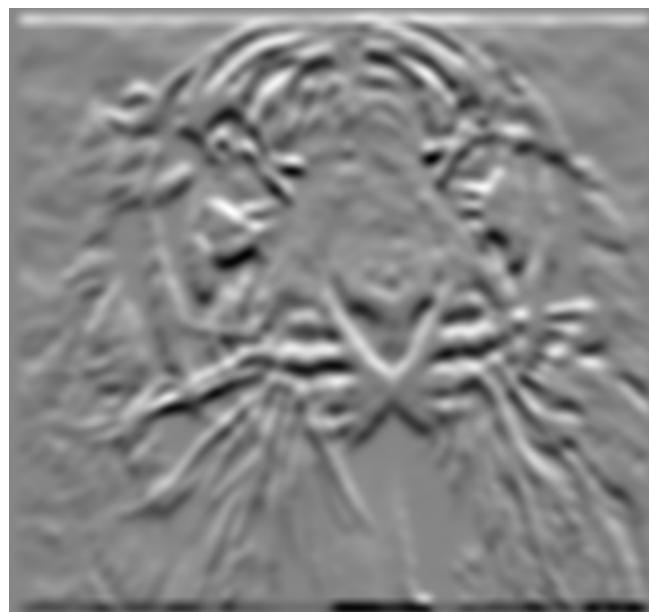


$$|\nabla I| = \sqrt{\frac{\partial I^2}{\partial x} + \frac{\partial I^2}{\partial y}}$$

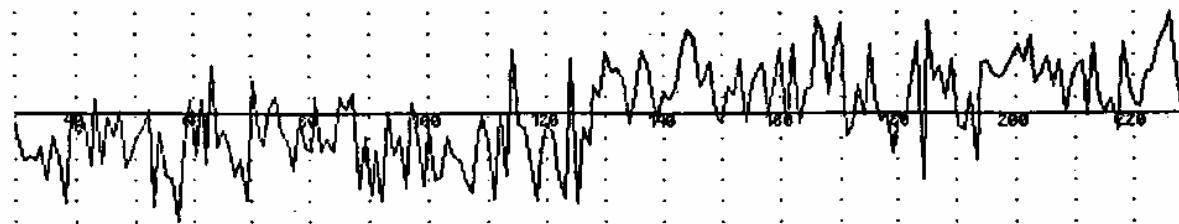
$\frac{\partial I}{\partial x}$



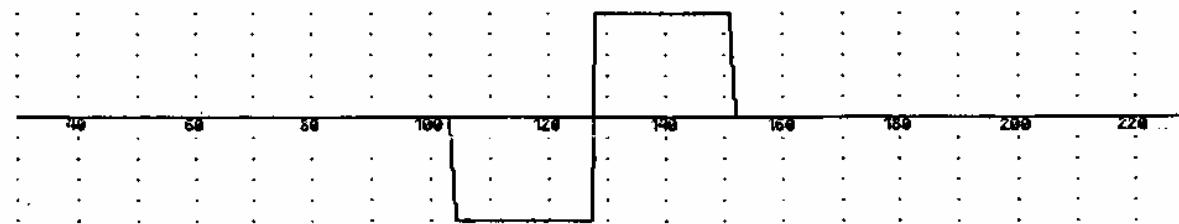
$\frac{\partial I}{\partial y}$



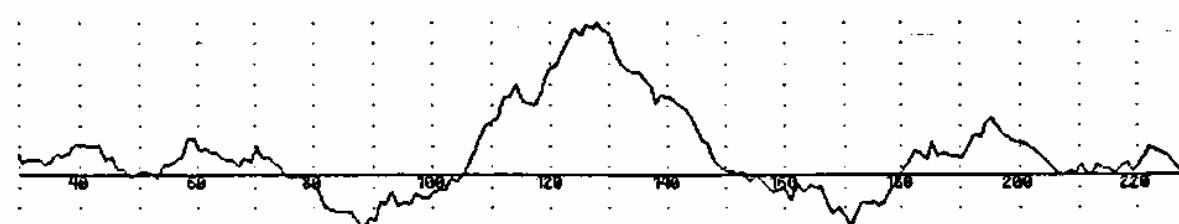
Input



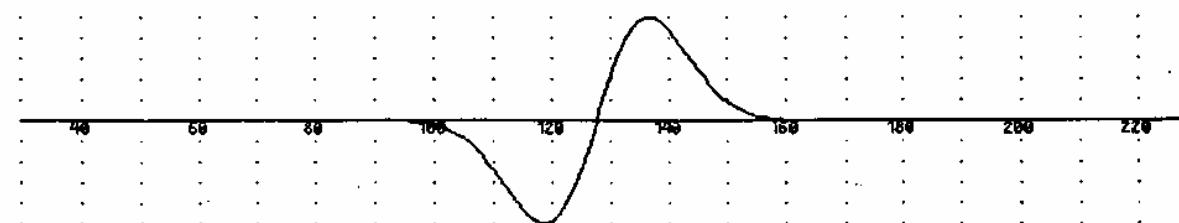
Difference operator



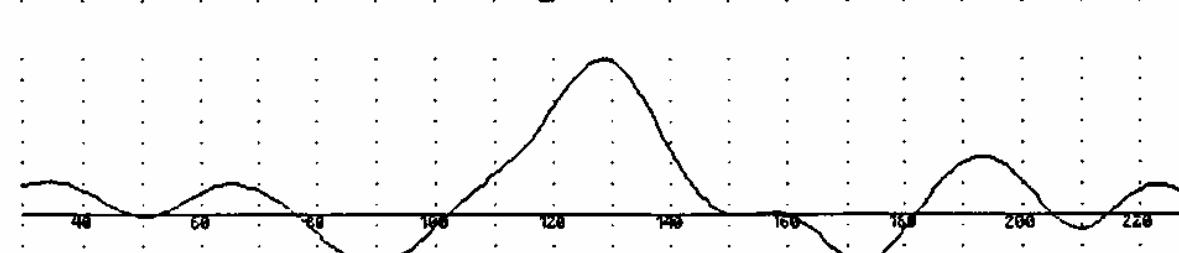
Derivative from  
difference operator



Gaussian derivative  
operator



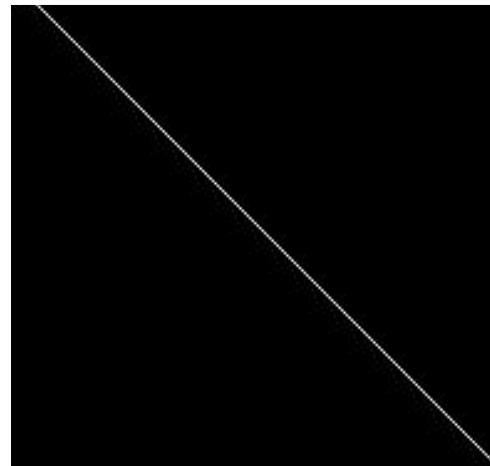
Derivative from  
Gaussian derivative



There is **ALWAYS** a tradeoff between smoothing and good edge localization!



Image with Edge



Edge Location

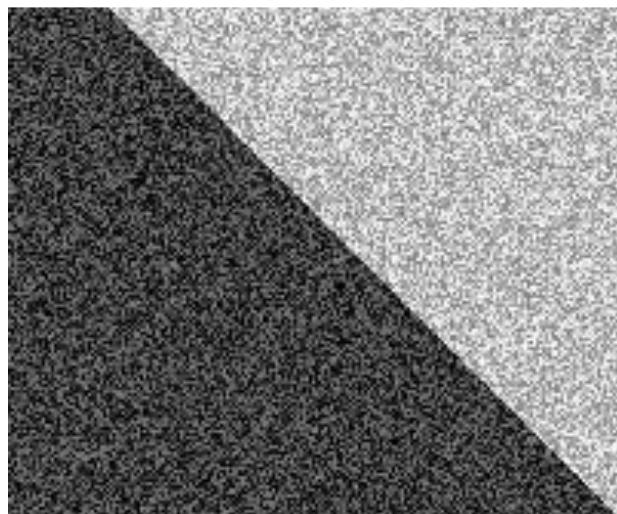
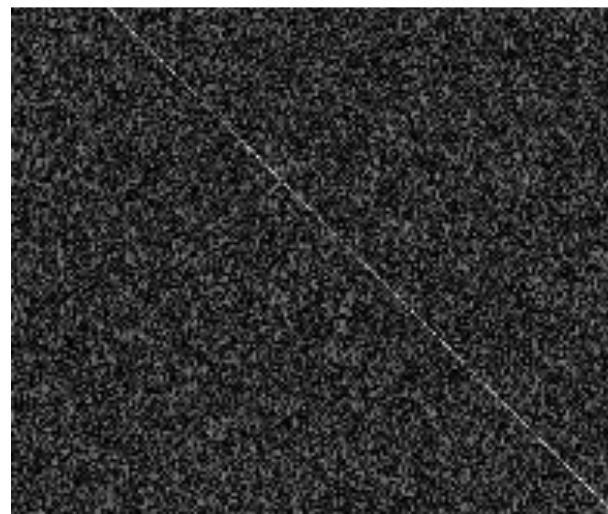
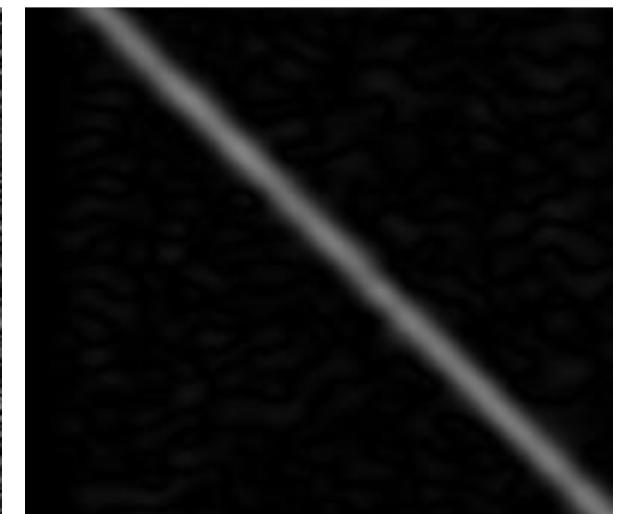


Image + Noise

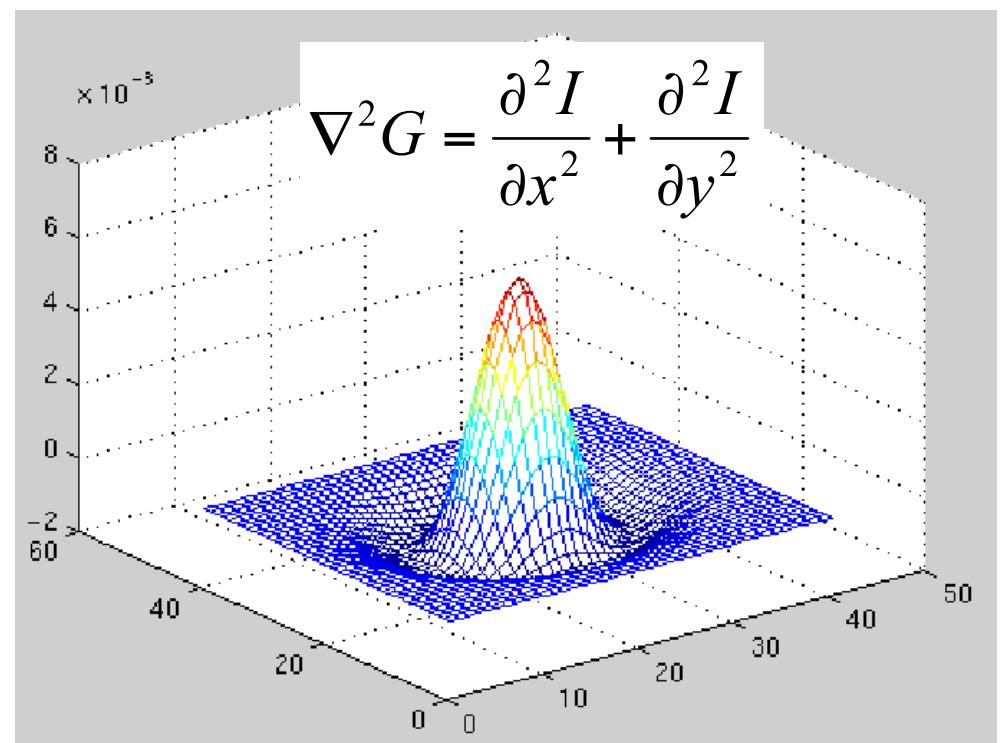
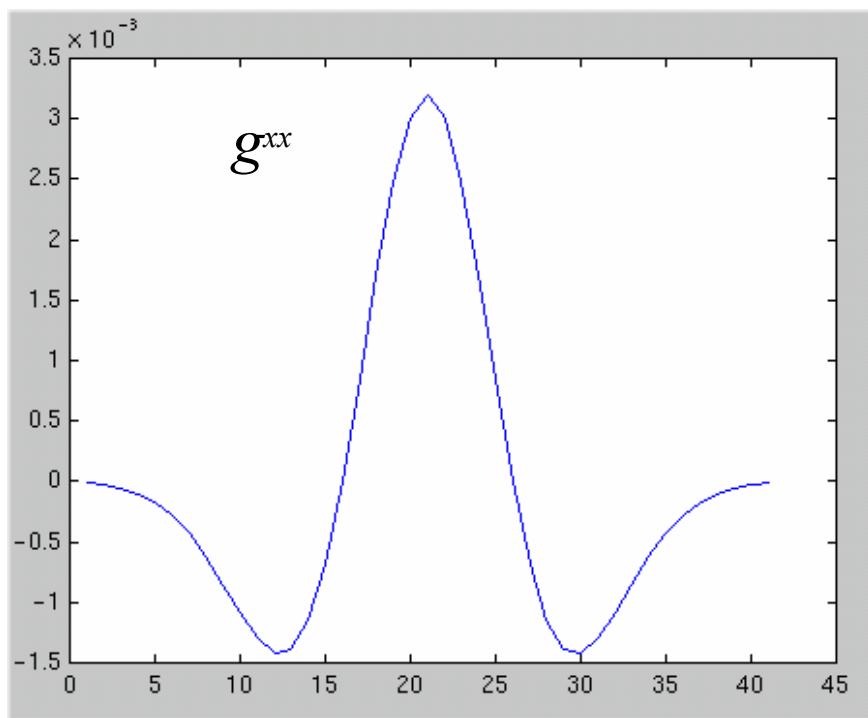
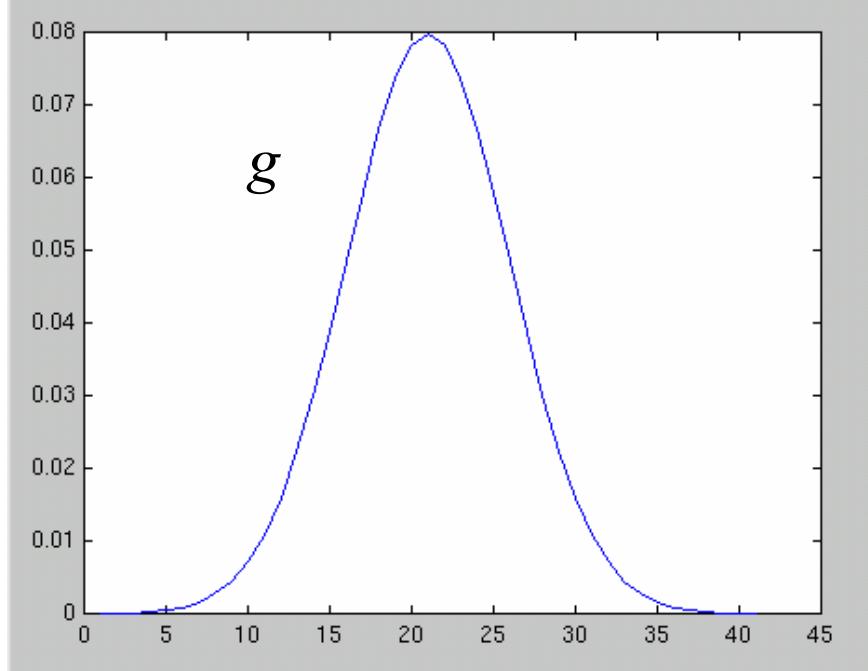


Derivatives detect  
edge *and* noise



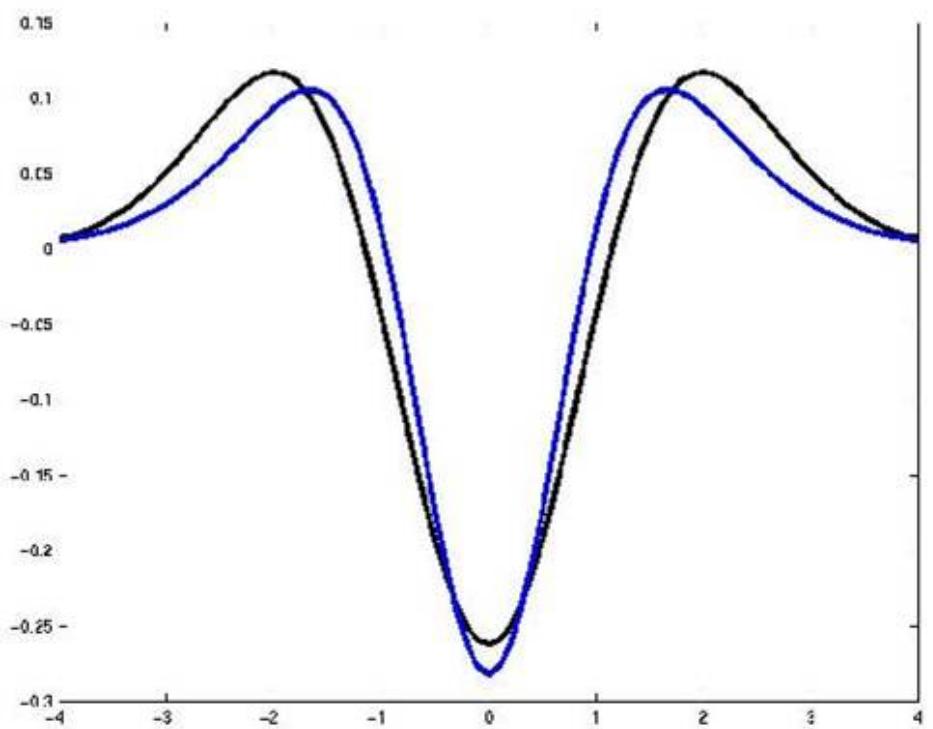
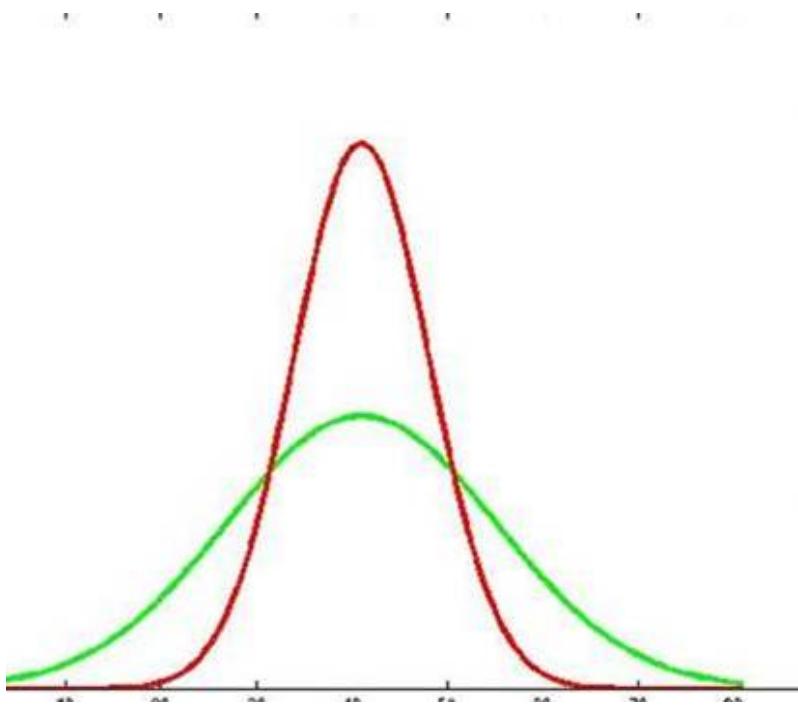
Smoothed derivative removes  
noise, but blurs edge

# Second derivatives: Laplacian



# DOG Approximation to LOG

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$



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

### Gaussian

Separable, low-pass filter

---

### Derivatives of Gaussian

$$\frac{\partial G_\sigma(x, y)}{\partial x} \propto x e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \frac{\partial G_\sigma(x, y)}{\partial y} \propto y e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Separable, output of convolution is gradient at scale  
 $\sigma$ :  $\nabla I = I * \nabla G_\sigma$

$$\nabla G_\sigma = \left[ \frac{\partial G_\sigma}{\partial x} \quad \frac{\partial G_\sigma}{\partial y} \right]^t$$

### Laplacian

$$\nabla^2 G_\sigma(x, y) = \frac{\partial^2 G_\sigma(x, y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x, y)}{\partial y^2}$$

Not-separable, approximated by A difference of Gaussians. Output of convolution is Laplacian of image: Zero-crossings correspond to edges

---

### Directional Derivatives

$$\cos \theta \frac{\partial G_\sigma}{\partial x} + \sin \theta \frac{\partial G_\sigma}{\partial y}$$

Output of convolution is magnitude of derivative in direction  $\theta$ . Filter is linear combination of derivatives in x and y

---

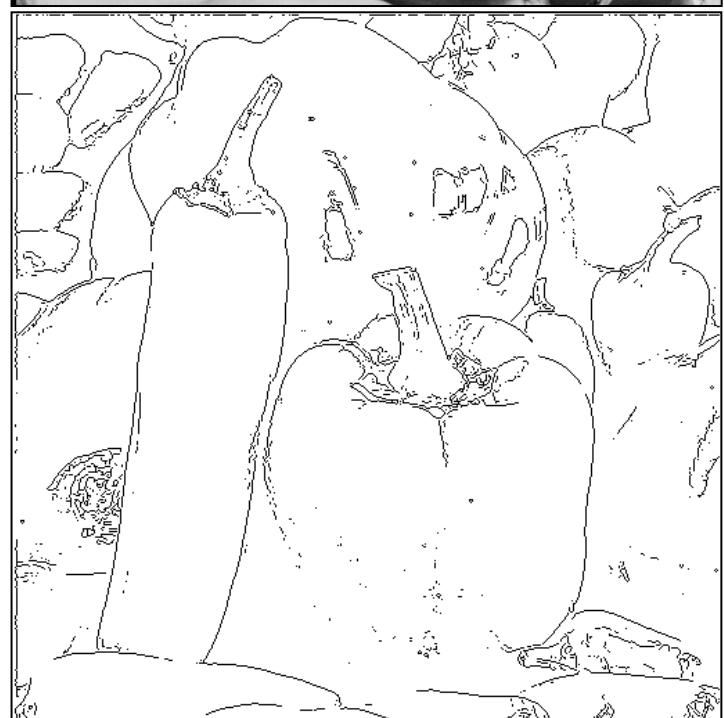
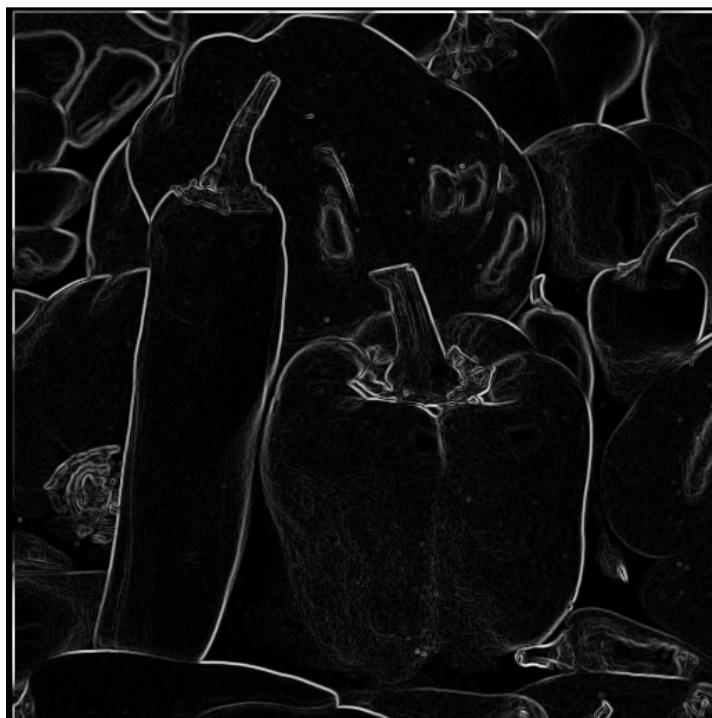
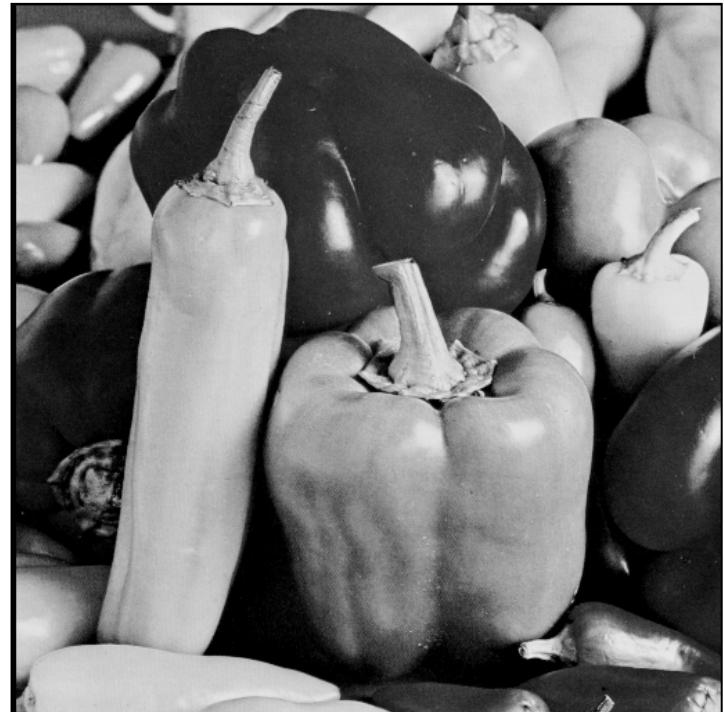
### Oriented Gaussian

$$e^{-\frac{(a_1x+b_1y)^2}{2\sigma_1^2} - \frac{(a_2x+b_2y)^2}{2\sigma_2^2}}$$

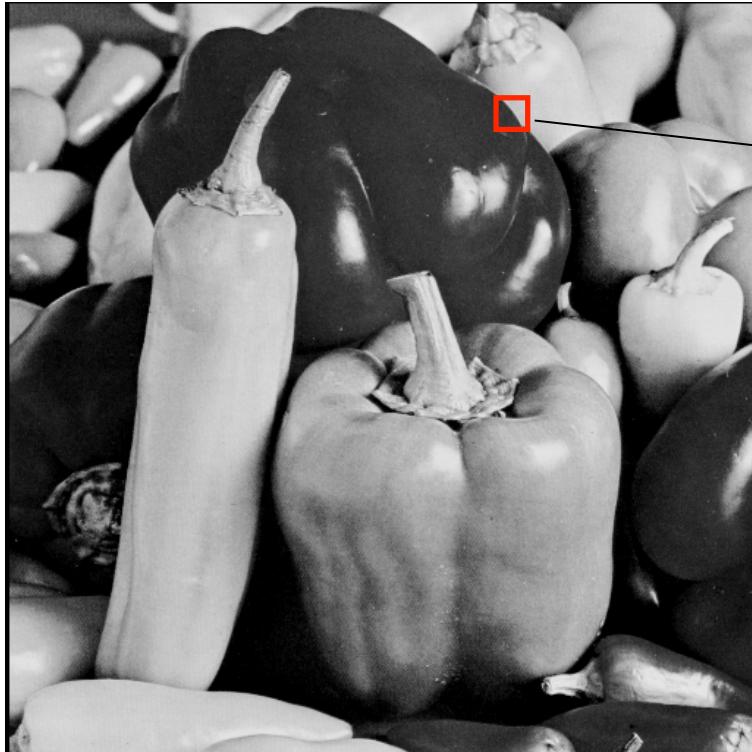
Smooth with different scales in orthogonal directions

# Edge Detection

- Edge Detection
  - Gradient operators
  - Canny edge detectors
  - Laplacian detectors

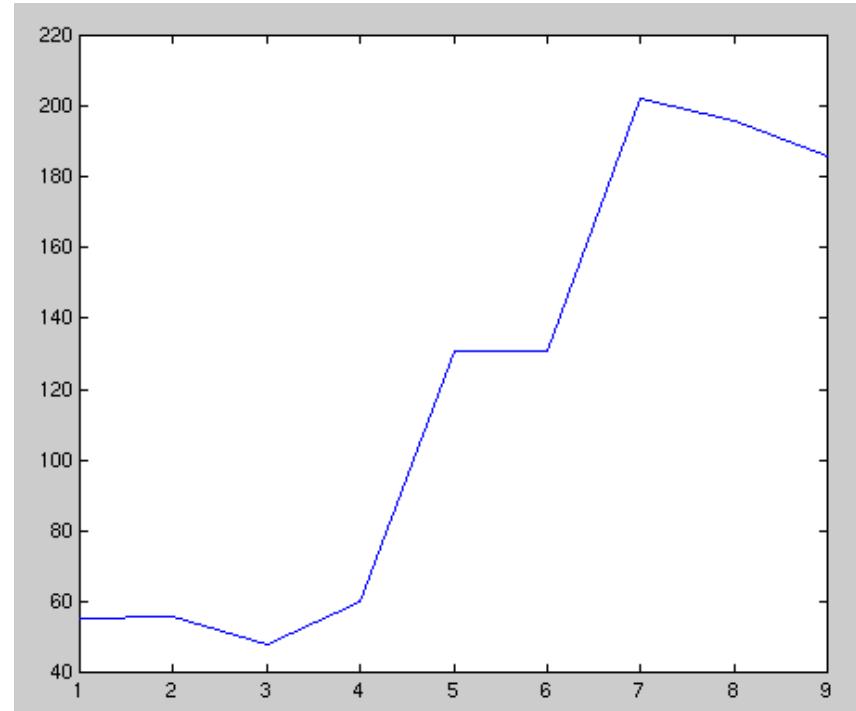


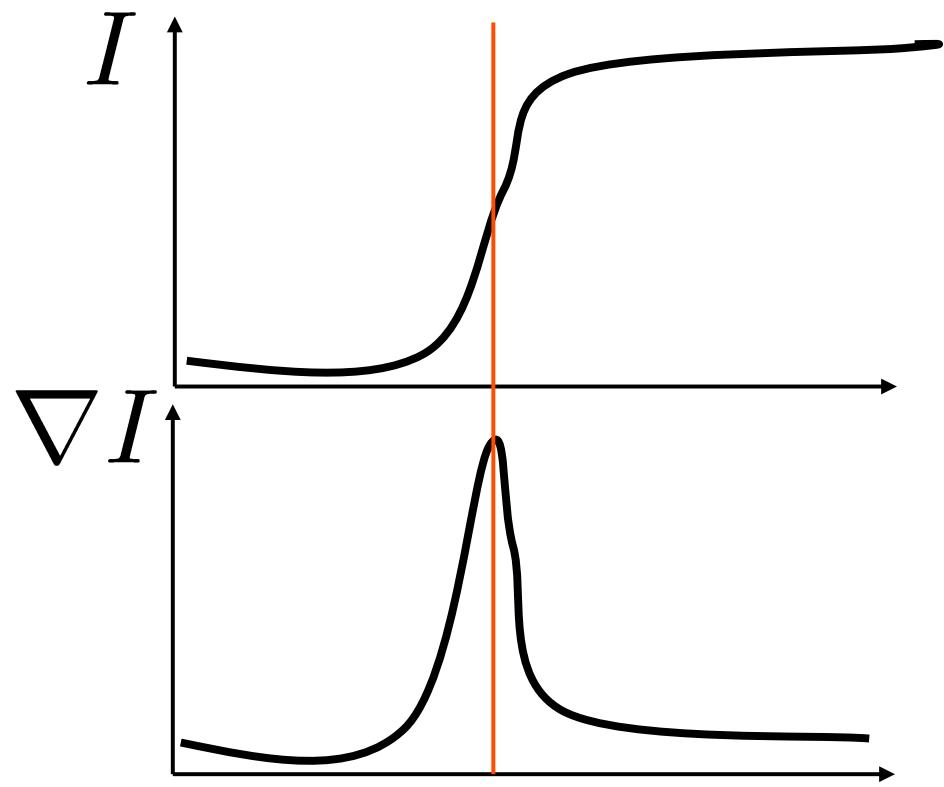
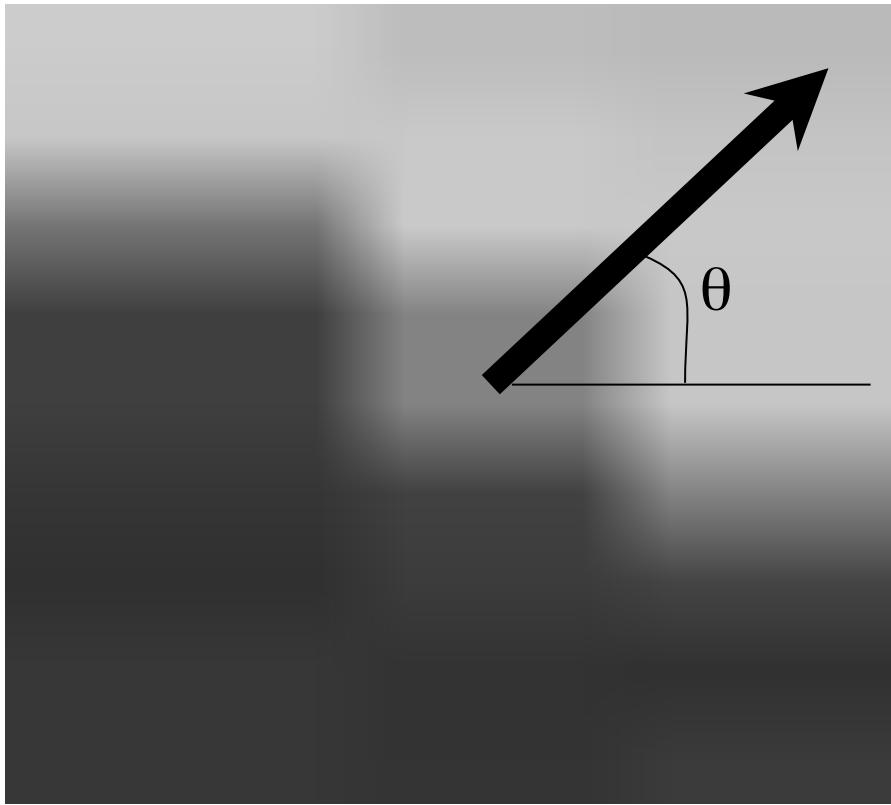
# What is an edge?



Edge = discontinuity of intensity in some direction.

Could be detected by looking for places where the derivatives of the image have large values.





Edge pixels are at local maxima of gradient magnitude  
 Gradient computed by convolution with Gaussian derivatives  
 Gradient direction is always perpendicular to edge direction

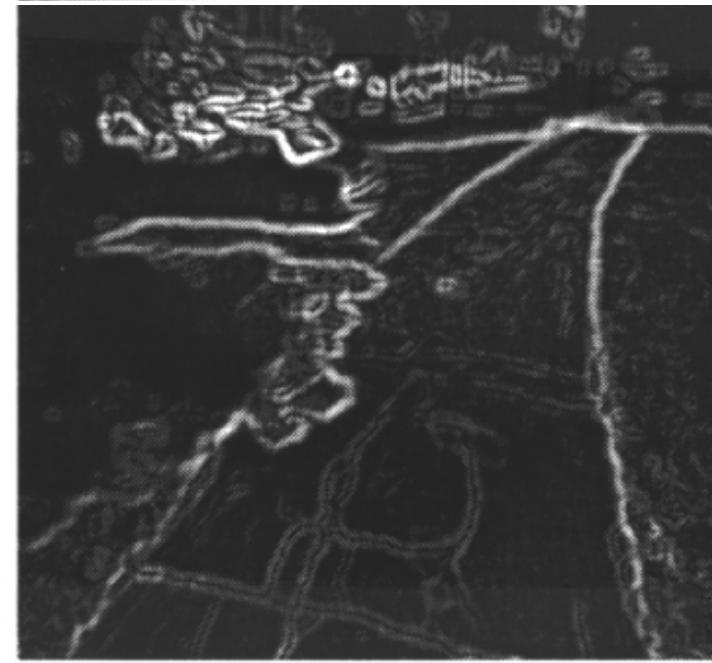
$$\frac{\partial I}{\partial x} = G_{\sigma}^x * I$$

$$\frac{\partial I}{\partial y} = G_{\sigma}^y * I$$

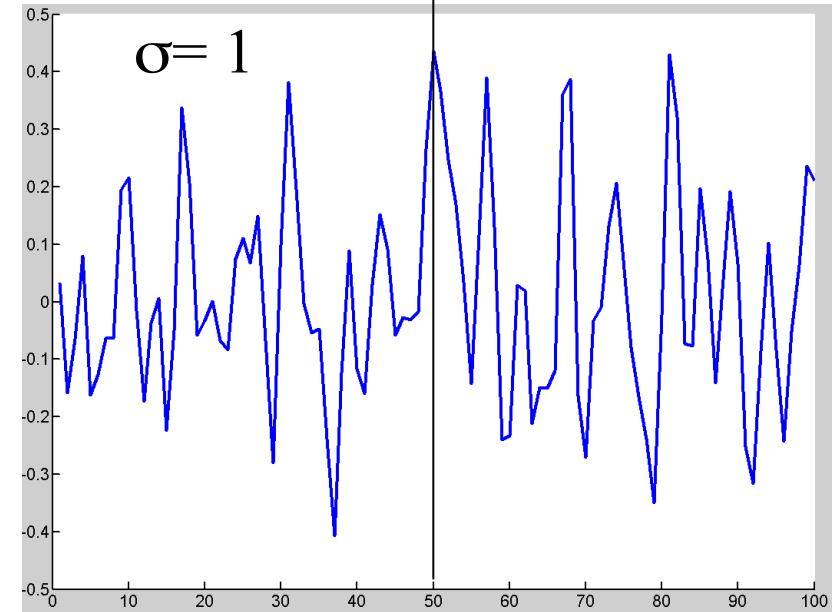
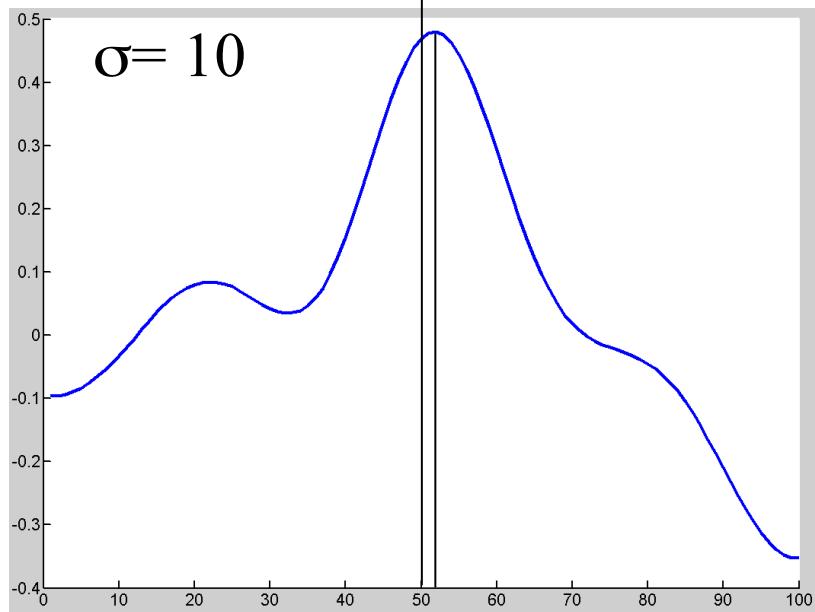
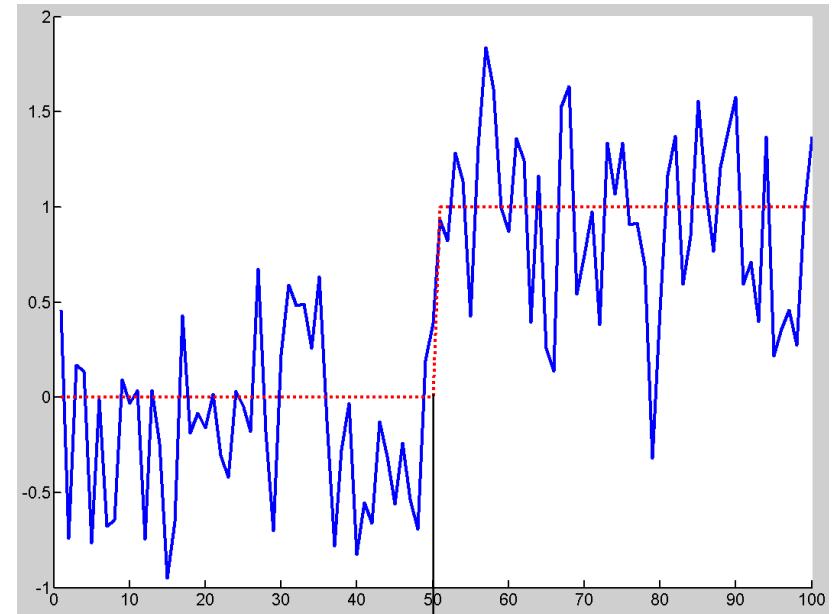
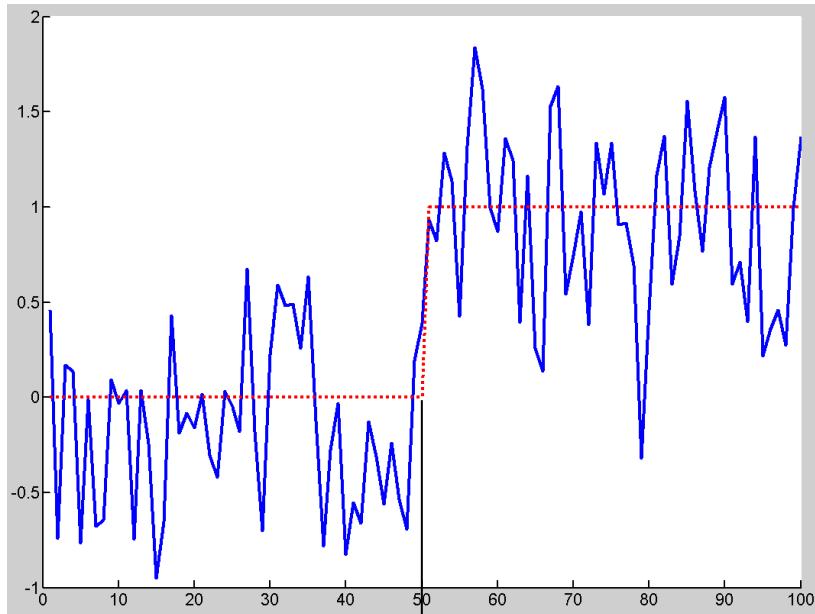
$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \quad \theta = \text{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$



Small sigma



Large sigma

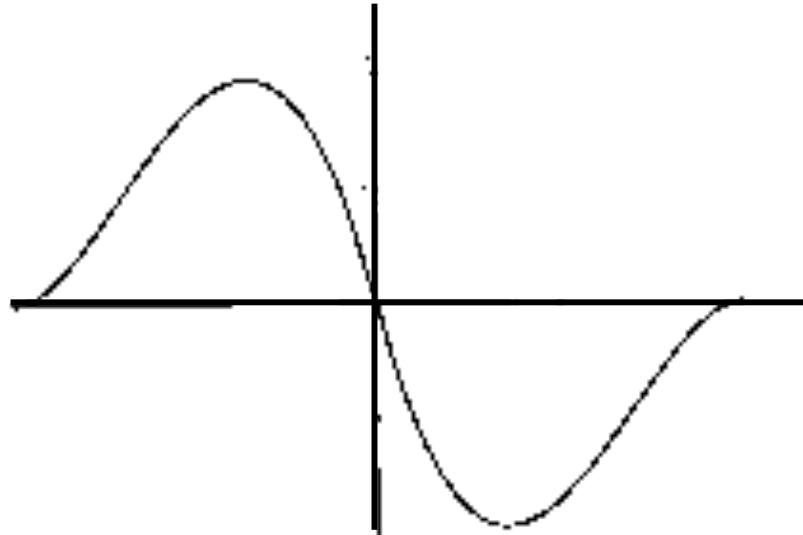


Large  $\sigma \rightarrow$  Good detection (high SNR)  
Poor localization

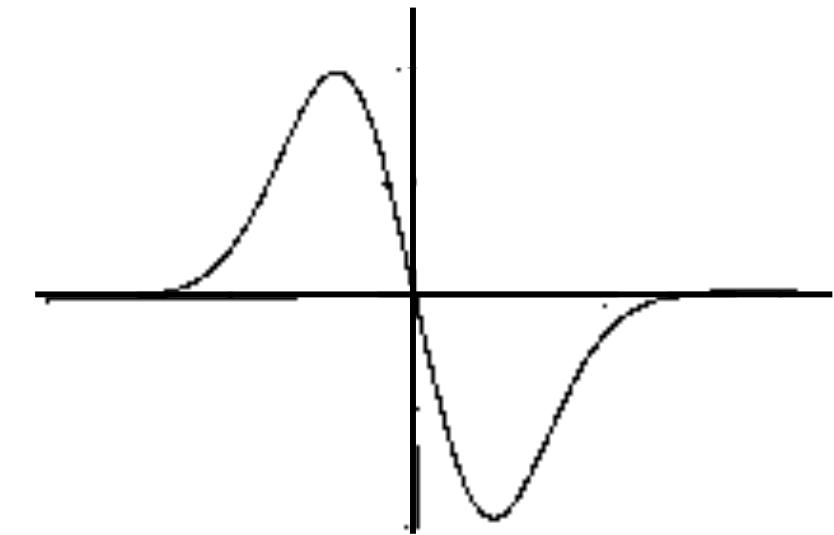
Small  $\sigma \rightarrow$  Poor detection (low SNR)  
Good localization

# Canny's Result

- Given a filter  $f$ , define the two objective functions:
  - $\Lambda(f)$  large if  $f$  produces good localization
  - $\Sigma(f)$  large if  $f$  produces good detection (high SNR)
- Problem: Find a family of filters  $f$  that maximizes the compromise criterion
$$\Lambda(f)\Sigma(f)$$
under the constraint that a single peak is generated by a step edge
- Solution: Unique solution, a close approximation is the Gaussian derivative filter!



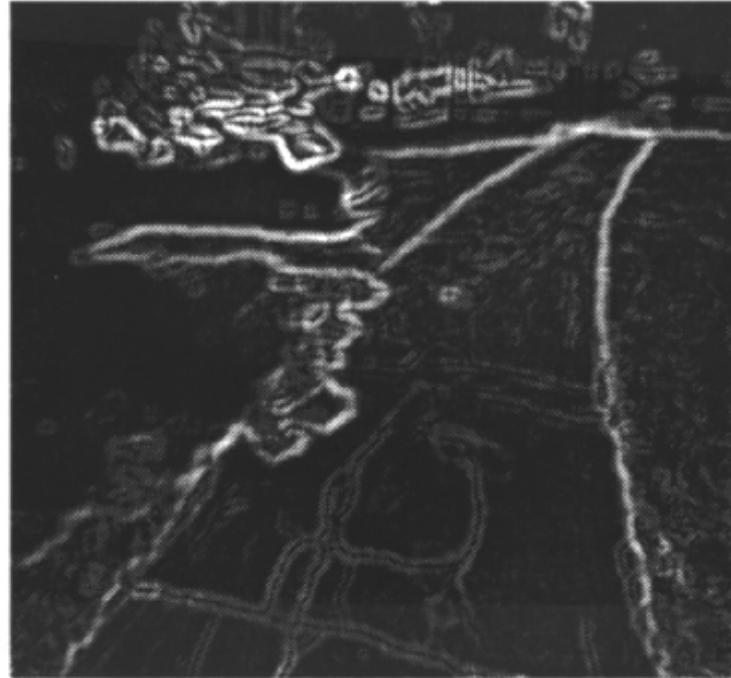
Canny

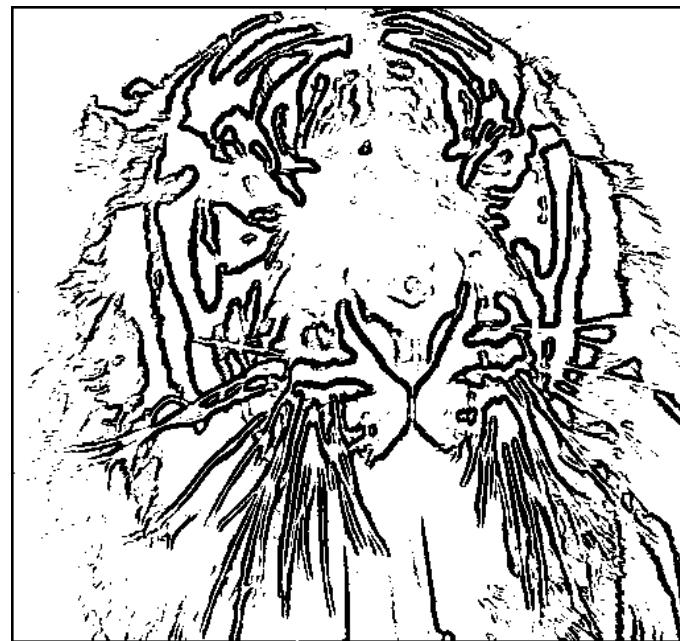


Derivative of Gaussian

# Next Steps

- The gradient magnitude enhances the edges but 2 problems remain:
  - What threshold should we use to retain only the “real” edges?
  - Even if we had a perfect threshold, we would still have poorly localized edges. How to extract optimally localize contours?
- Solution: Two standard tools:
  - Non-local maxima suppression
  - Hysteresis thresholding





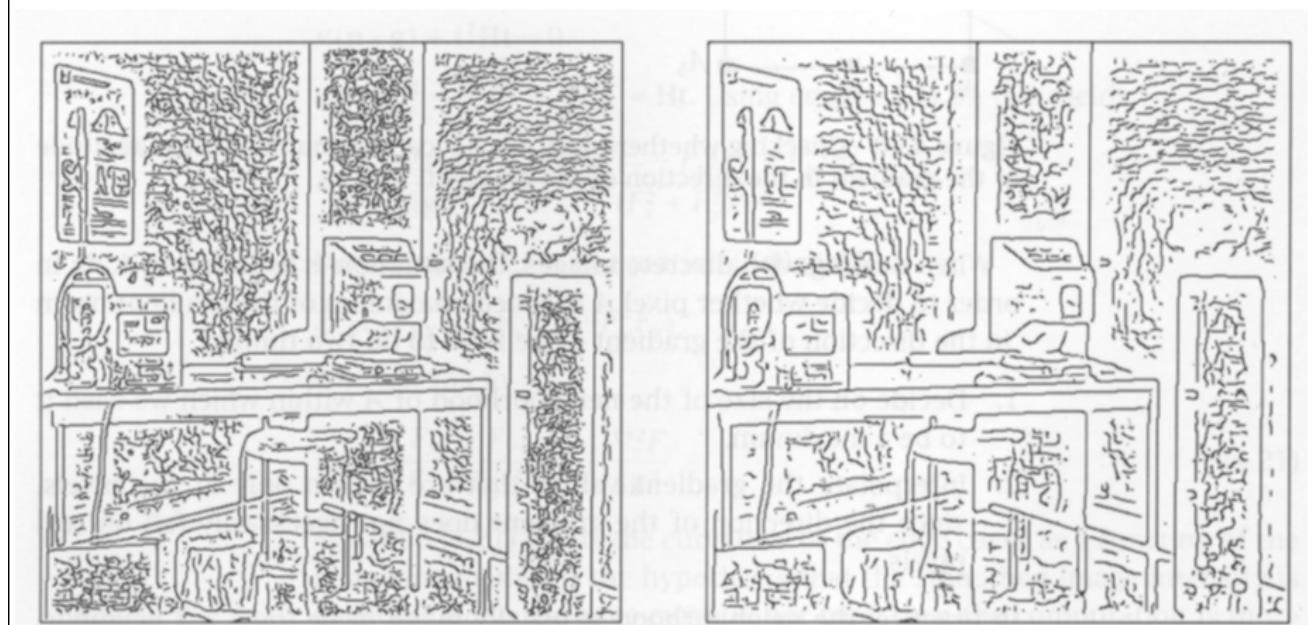
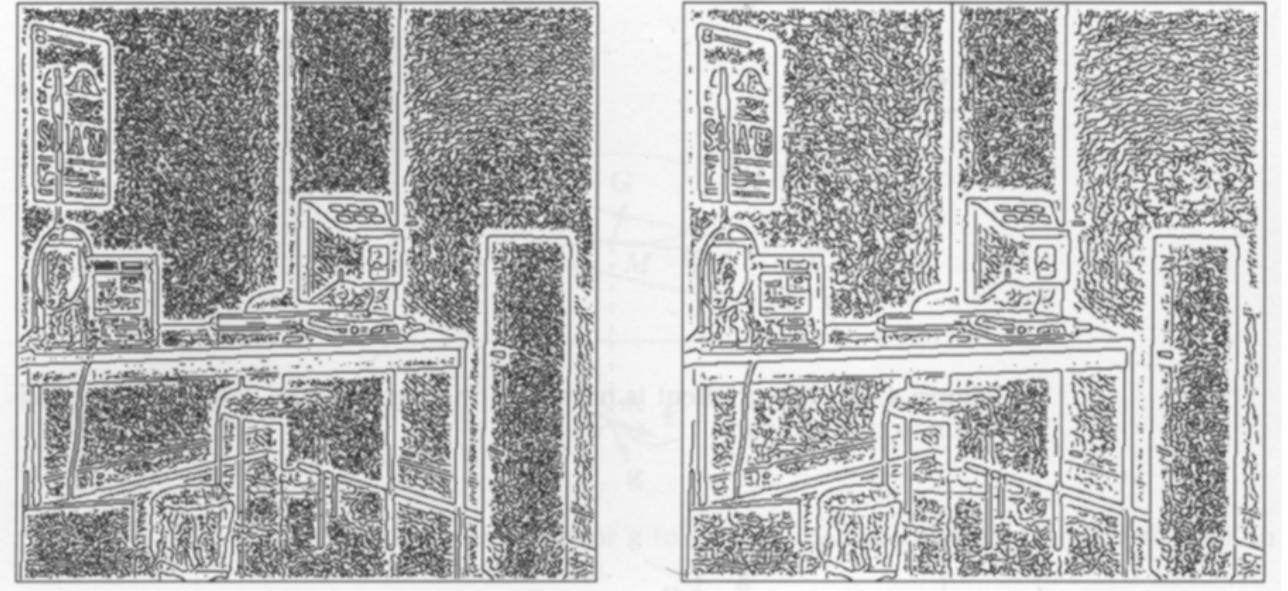
Different thresholds  
applied to gradient  
magnitude



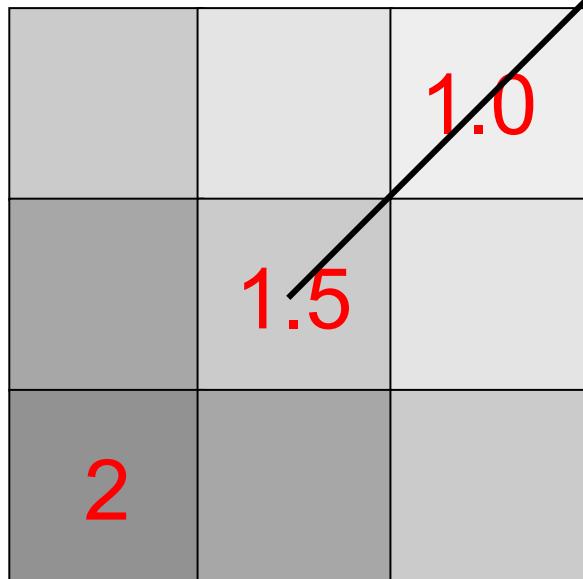
Input image



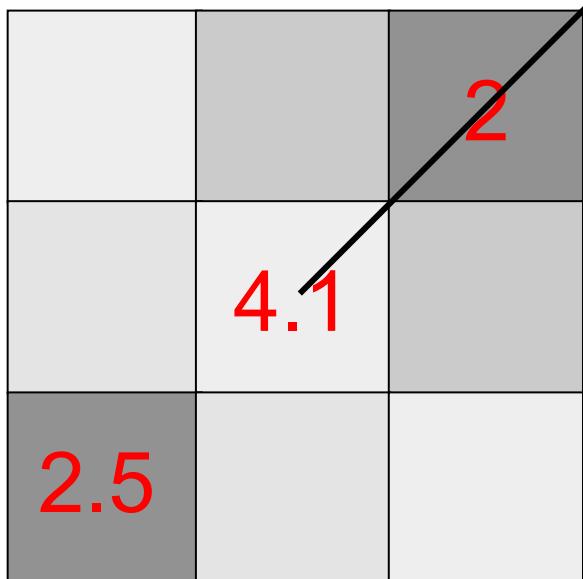
Different thresholds applied to gradient magnitude



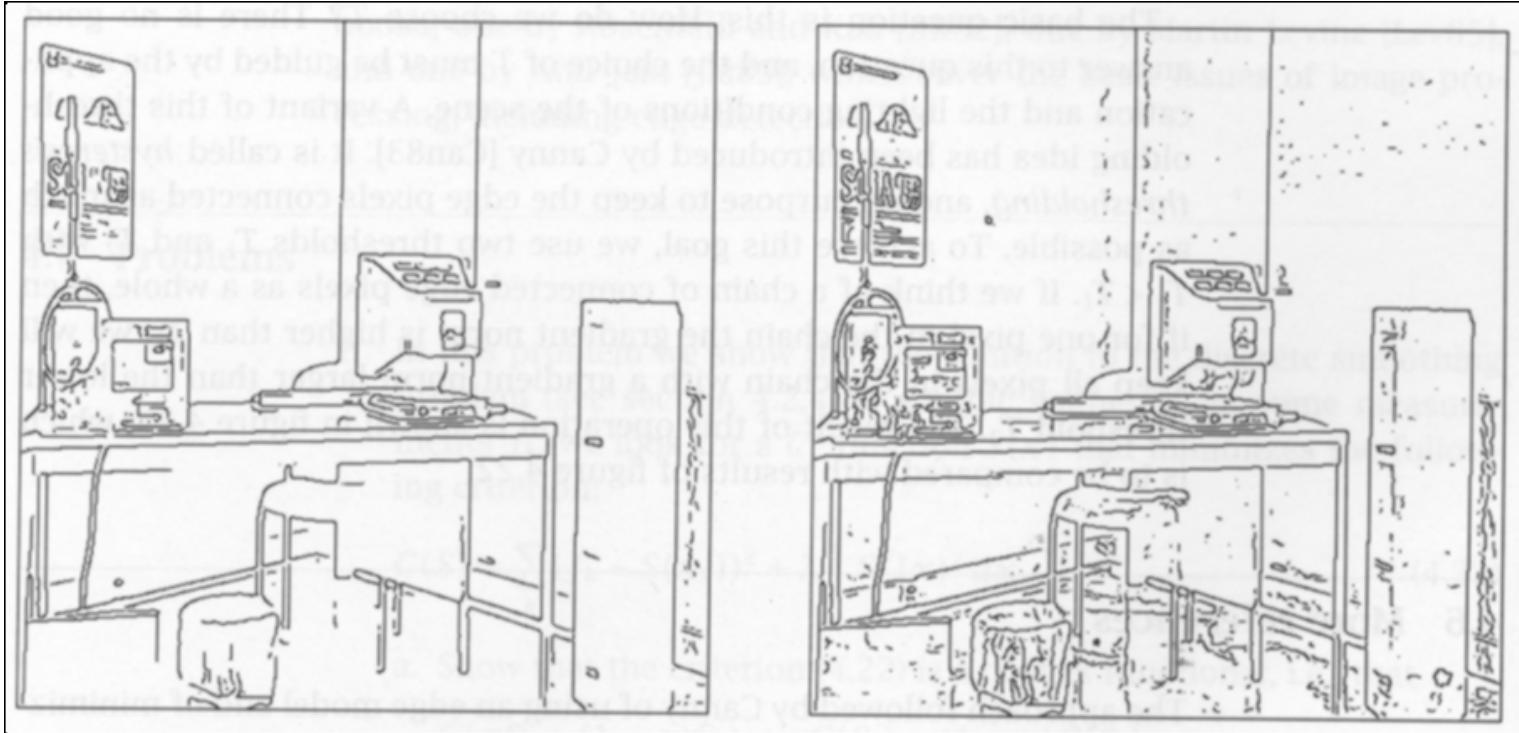
# Non-Local Maxima Suppression



Gradient magnitude at center pixel  
is lower than the gradient magnitude  
of a neighbor *in the direction of the gradient*  
→ Discard center pixel (set magnitude to 0)



Gradient magnitude at center pixel  
is greater than gradient magnitude  
of all the neighbors *in the direction  
of the gradient*  
→ Keep center pixel unchanged



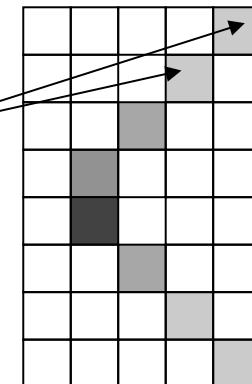
$T = 15$

$T = 5$

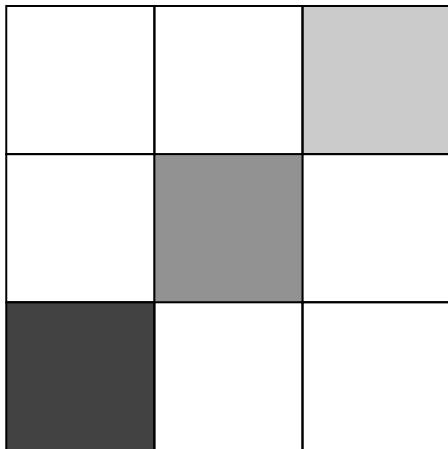
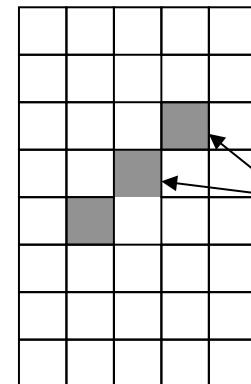
Two thresholds applied to gradient magnitude

# Hysteresis Thresholding

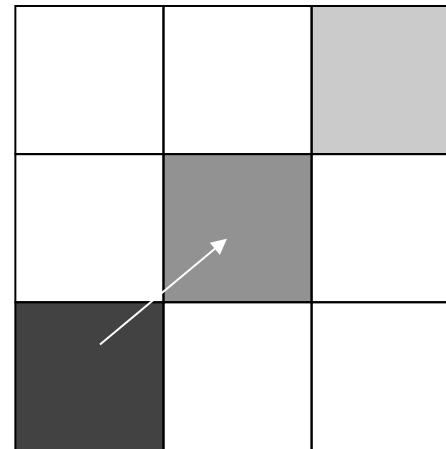
Weak pixels but connected



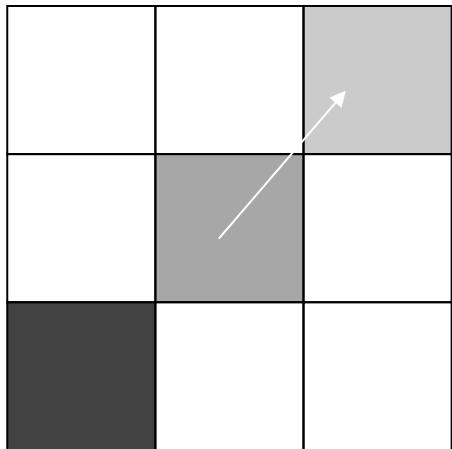
Weak pixels but isolated



Very strong edge response.  
Let's start here



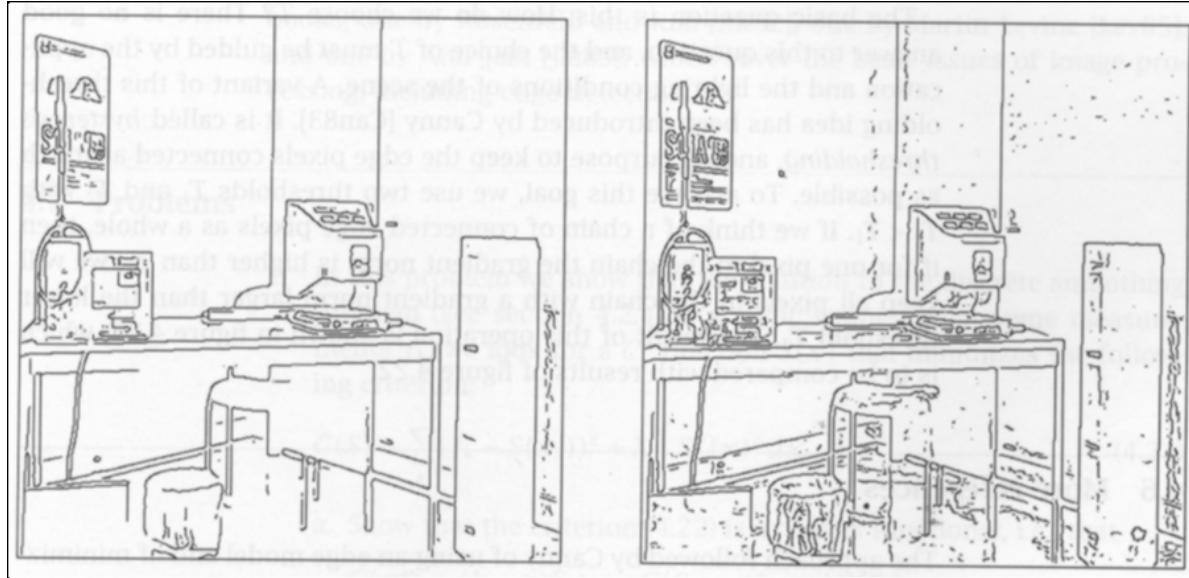
Weaker response but it is  
connected to a confirmed  
edge point. Let's keep it.



Continue....

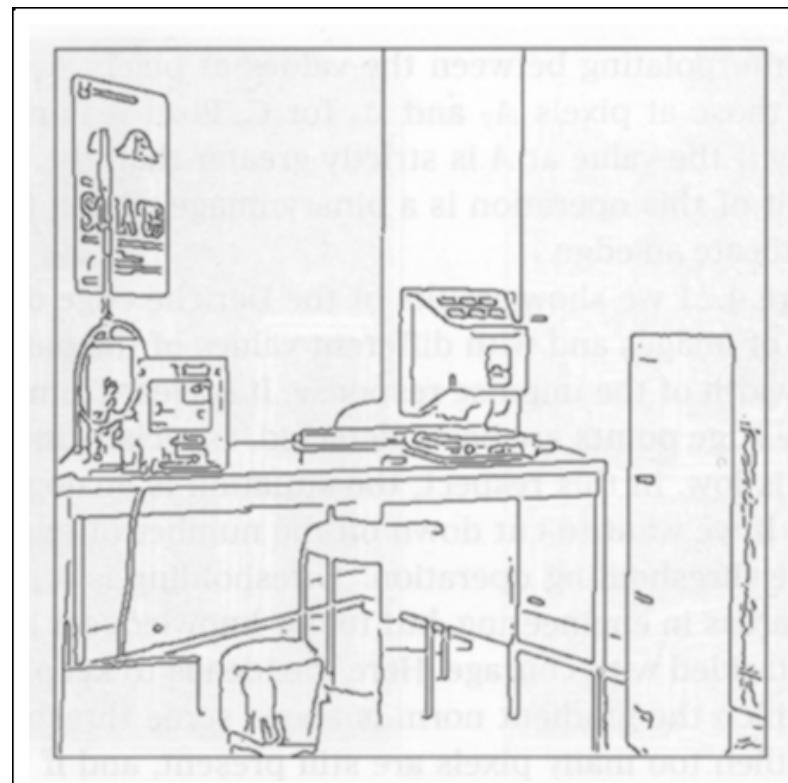
Note: Darker squares illustrate stronger edge response (larger  $M$ )

$T=15$



$T=5$

Hysteresis  
thresholding



Hysteresis  
 $T_h=15 \quad T_l = 5$

1. Compute  $x$  and  $y$  derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x, y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding

- Select the pixels such that  $M > T_h$  (high threshold)
- Collect the pixels such that  $M > T_l$  (low threshold) that are neighbors of already collected edge points

# Summary

- Edges are discontinuities of intensity in images
- Correspond to local maxima of image gradient
- Gradient computed by convolution with derivatives of Gaussian
- General principle applies:
  - Large  $\sigma$ : Poor localization, good detection
  - Small  $\sigma$ : Good localization, poor detection
- Canny showed that Gaussian derivatives yield good compromise between localization and detection
- Edges correspond to zero-crossings of the second derivative (Laplacian in 2-D)