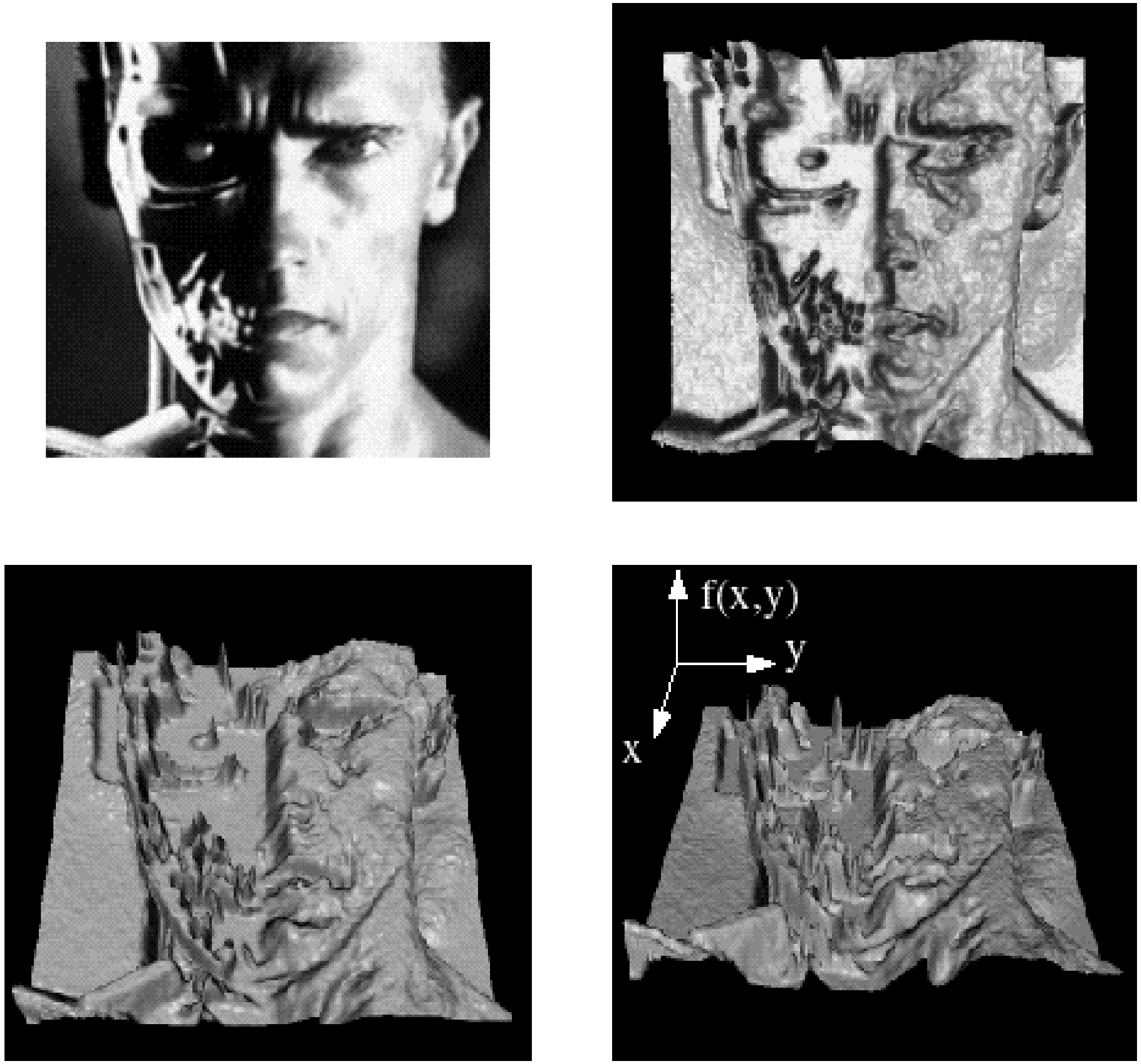
Filtering, Edge Detection, Frequency Analysis, Interest Point Operators

Handout I

Forsyth&Ponce: 7,8,9

Szeliski: 3,4

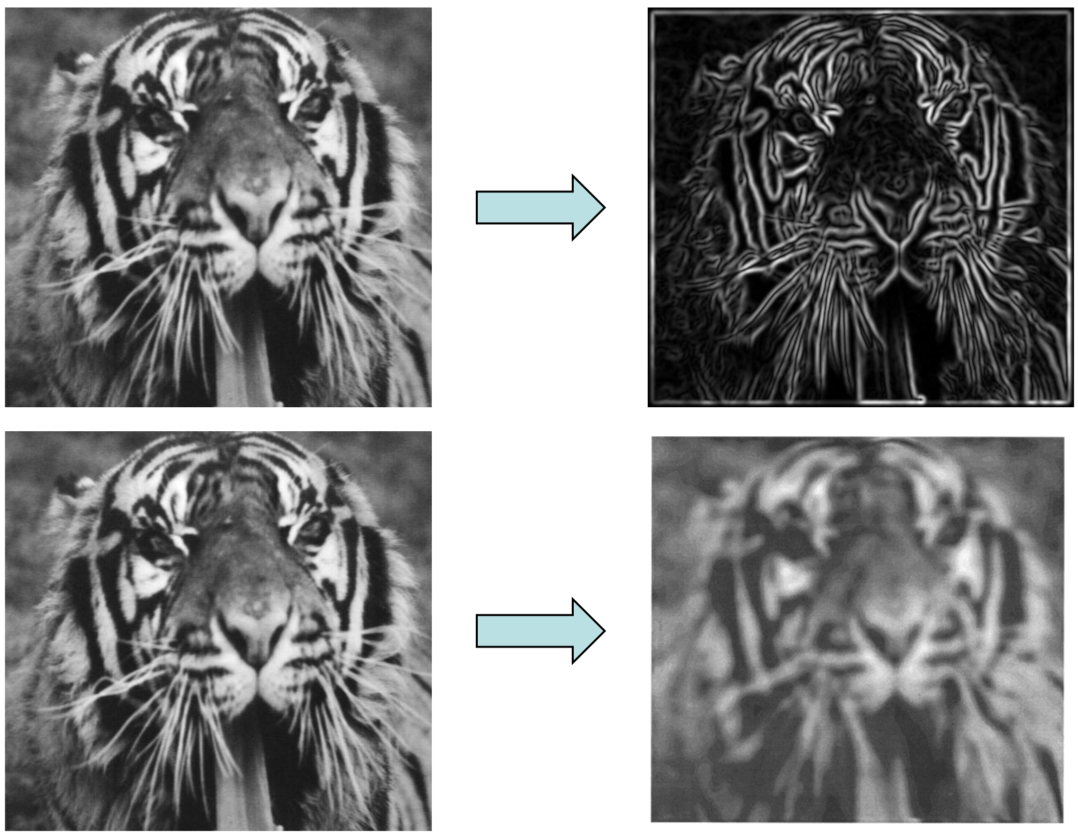
- Basic Filters
  - Convolution/Correlation/Linear filtering
  - Gaussian filters
  - Smoothing and noise reduction
  - First derivatives of Gaussian
  - Second derivative of Gaussian: Laplacian
  - Oriented Gaussian filters
  - Steerability
- Edge Detection
  - Gradient operators
  - Canny edge detectors
- Frequency analysis:
  - Fourier Transform
  - Gabor filters
  - Wavelets
  - Image pyramids
- Example Application:
  - Texture classification
- Interest point detection
  - Shape matrix
  - Harris detector
  - Scale invariance
-



- Image interpreted either as:
  - Continuous function $f(x,y)$
  - Discrete array $f[x,y]$

Basic Filters

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



Correlation

- 1D Formula:

$$\left(h \otimes f\right)(x) = \int_u h(x+u) f(u) du$$

$$\left(h \otimes f\right)[x] = \sum_i h[x+i] f[i]$$

- 2D Formula:

$$\left(h \otimes f\right)(x,y) = \int_u h(x+u, y+v) f(u,v) du$$

$$\left(h \otimes f\right)[x,y] = \sum_i h[x+i, y+j] f[i,j]$$

- Example on the web:
  - http://www.jhu.edu/~signals/convolve/

Convolution

- 1D Formula:

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

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

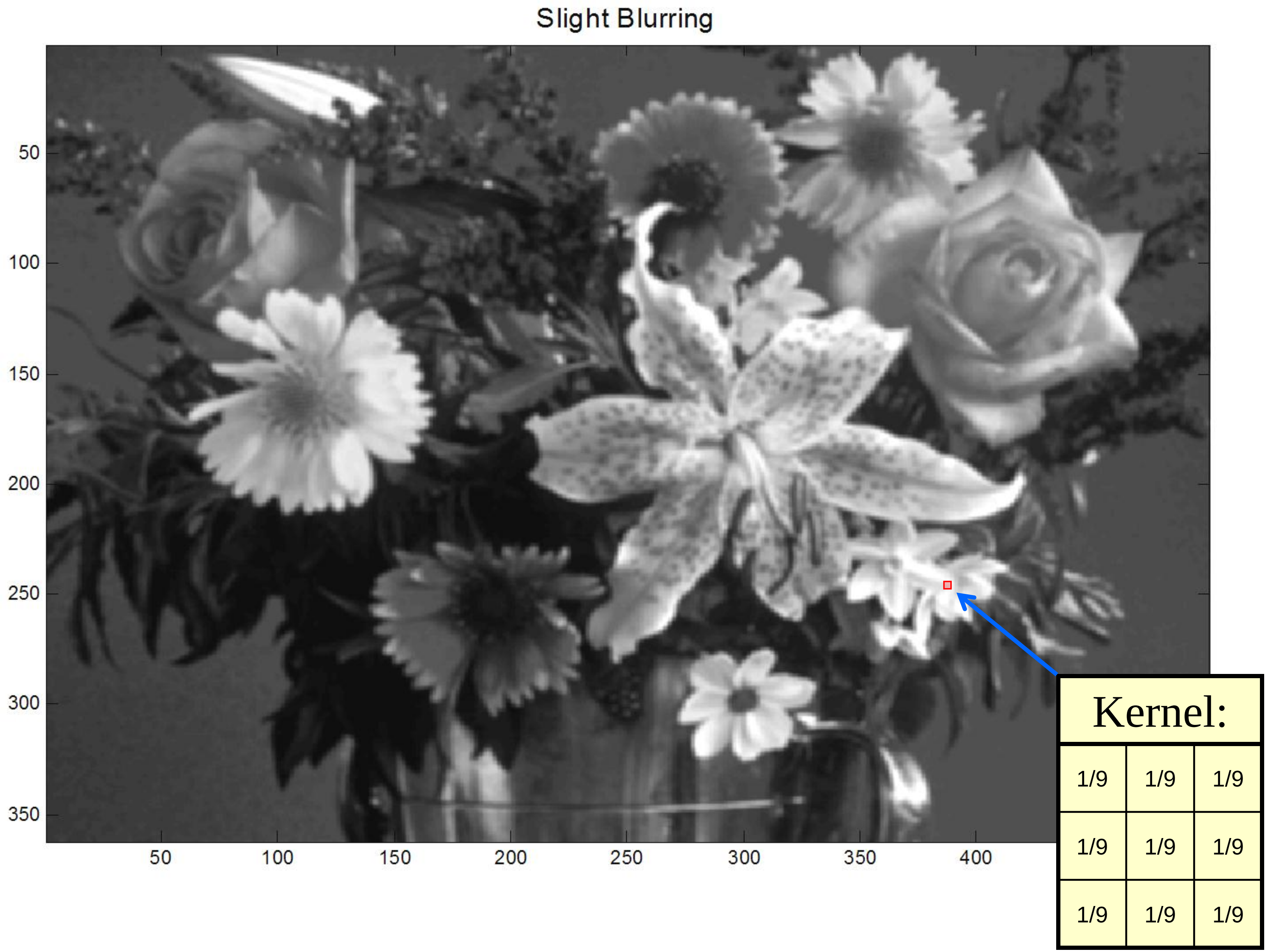
- 2D Formula:

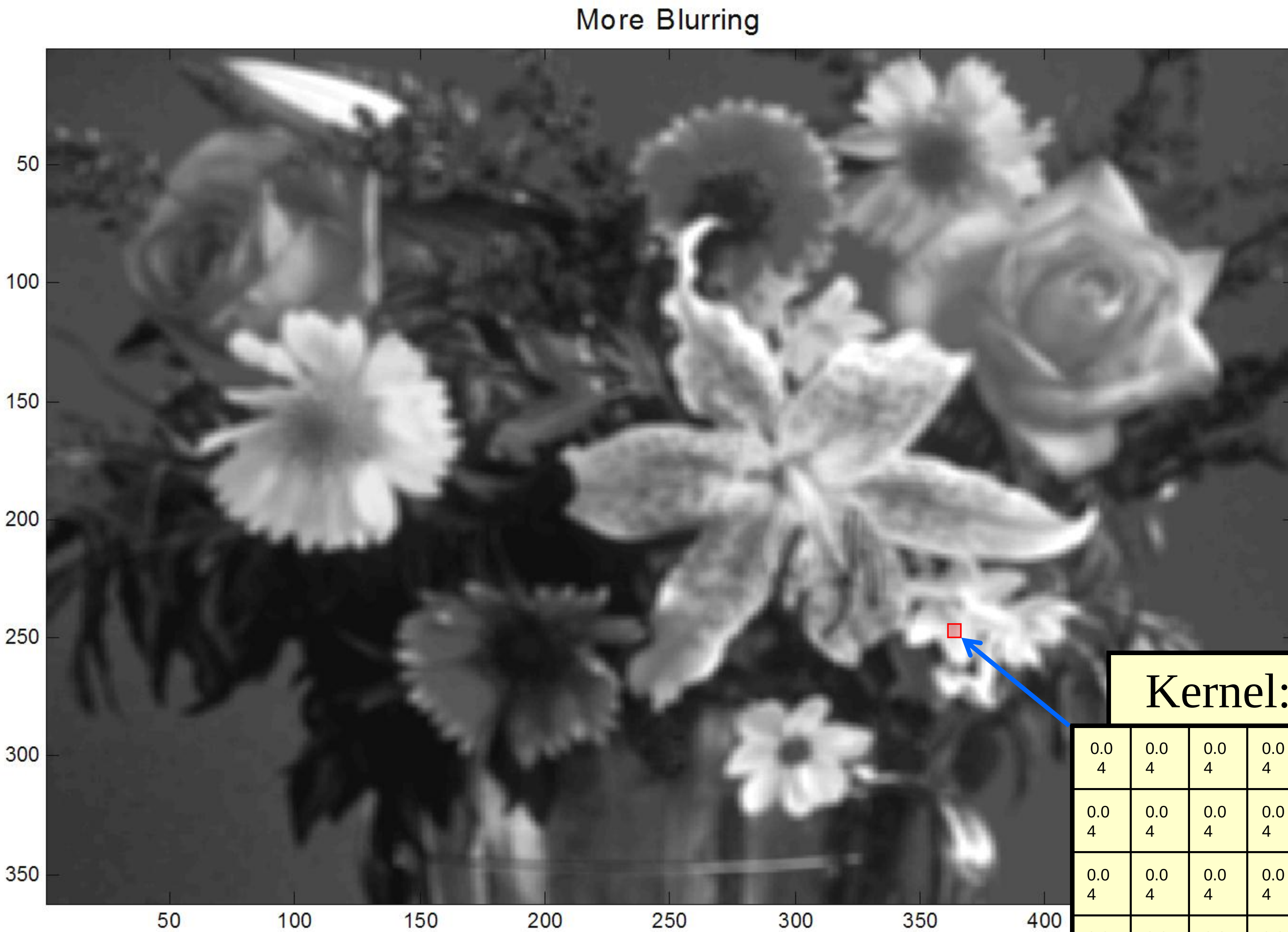
$$(h*f)(x,y) = \int_u h(x-u,y-v)f(u,v)du$$

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

- Example on the web:
  - http://www.jhu.edu/~signals/convolve/

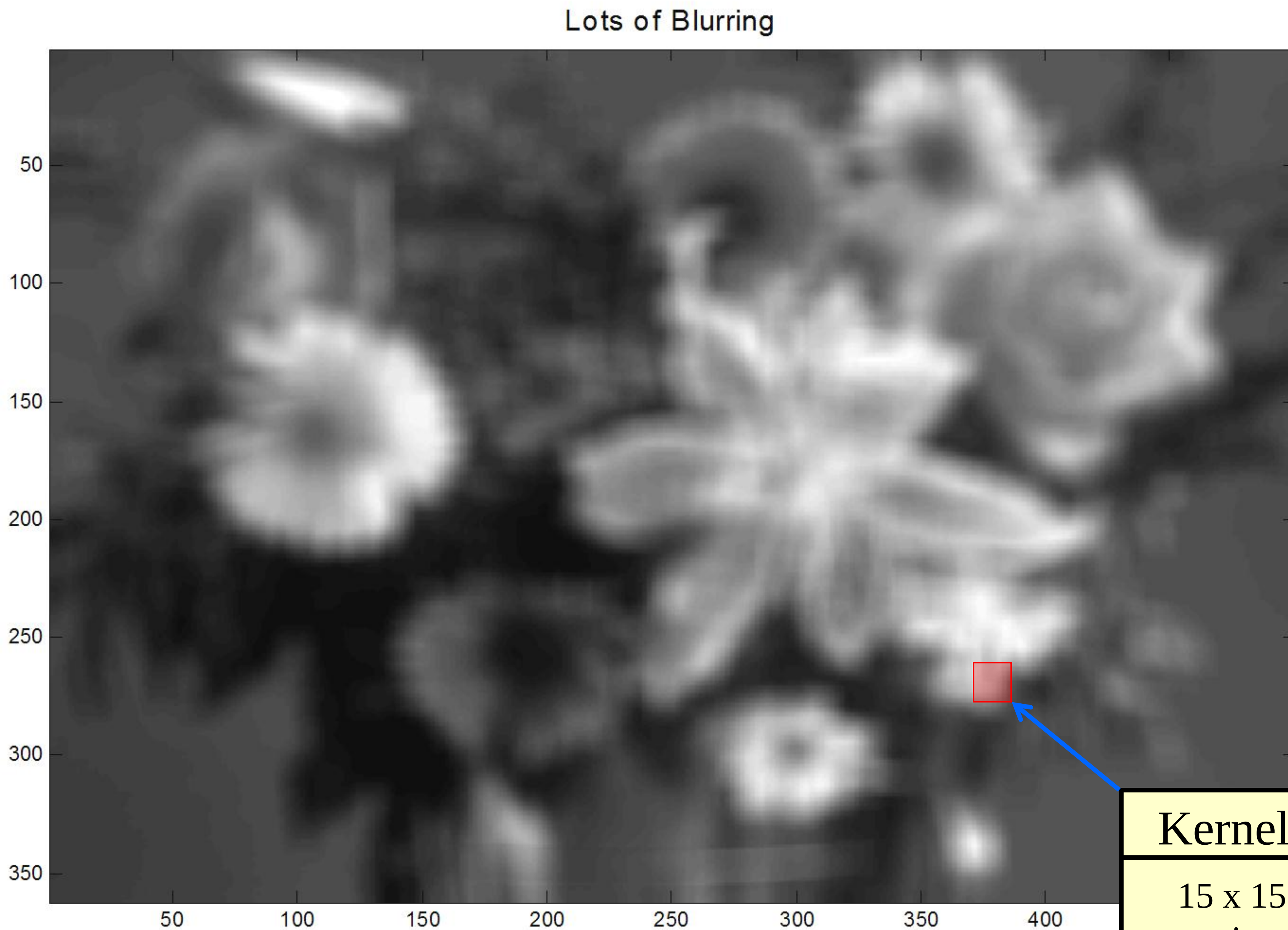

Original Image
50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500


Slight Blurring
50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
Kernel:
1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9



Kernel:

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



Kernel:

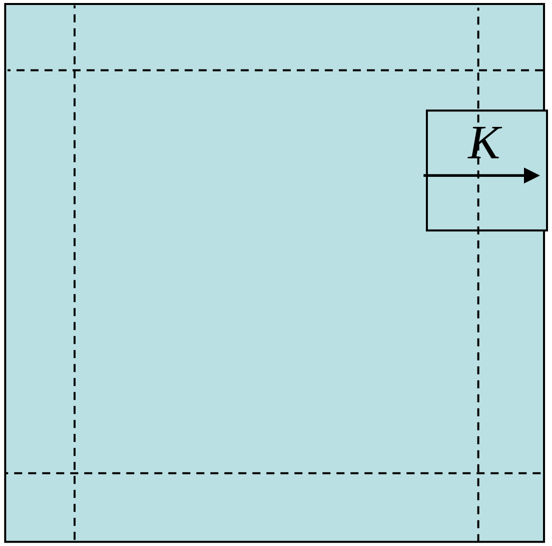
15 x 15 matrix of value 1/225

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: $$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

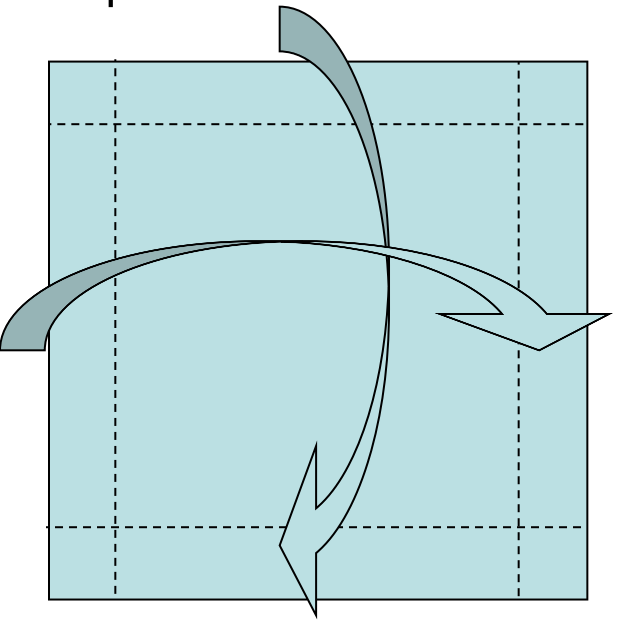
Practicalities (discrete convolution/correlation

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

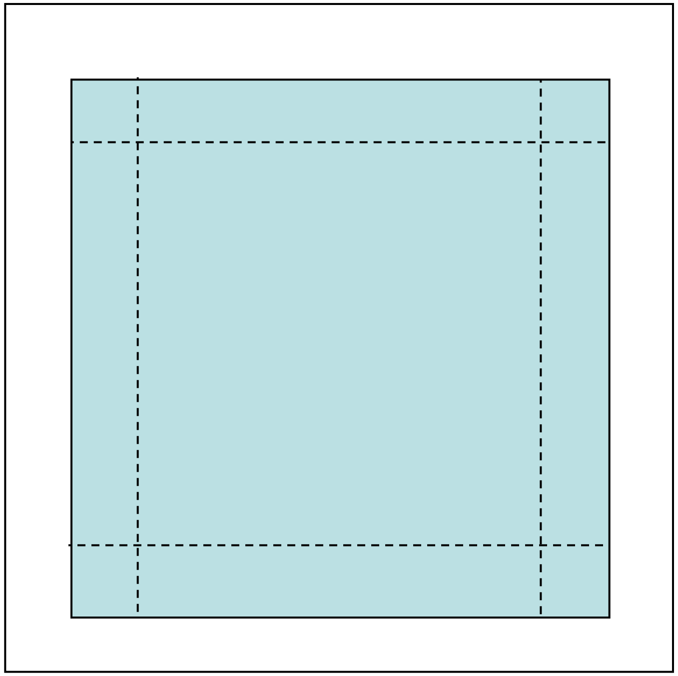


- Options:

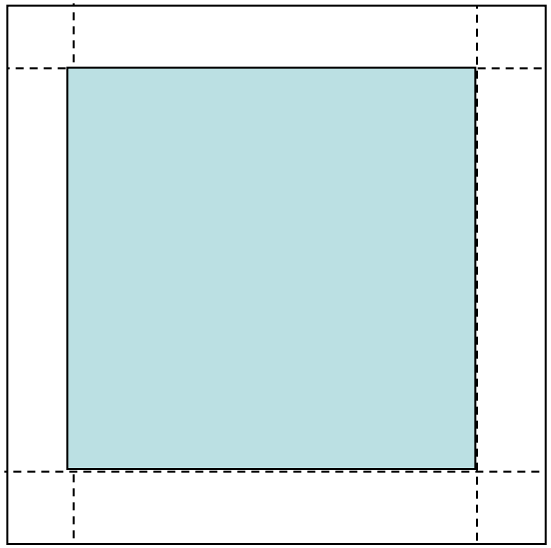
Warp around



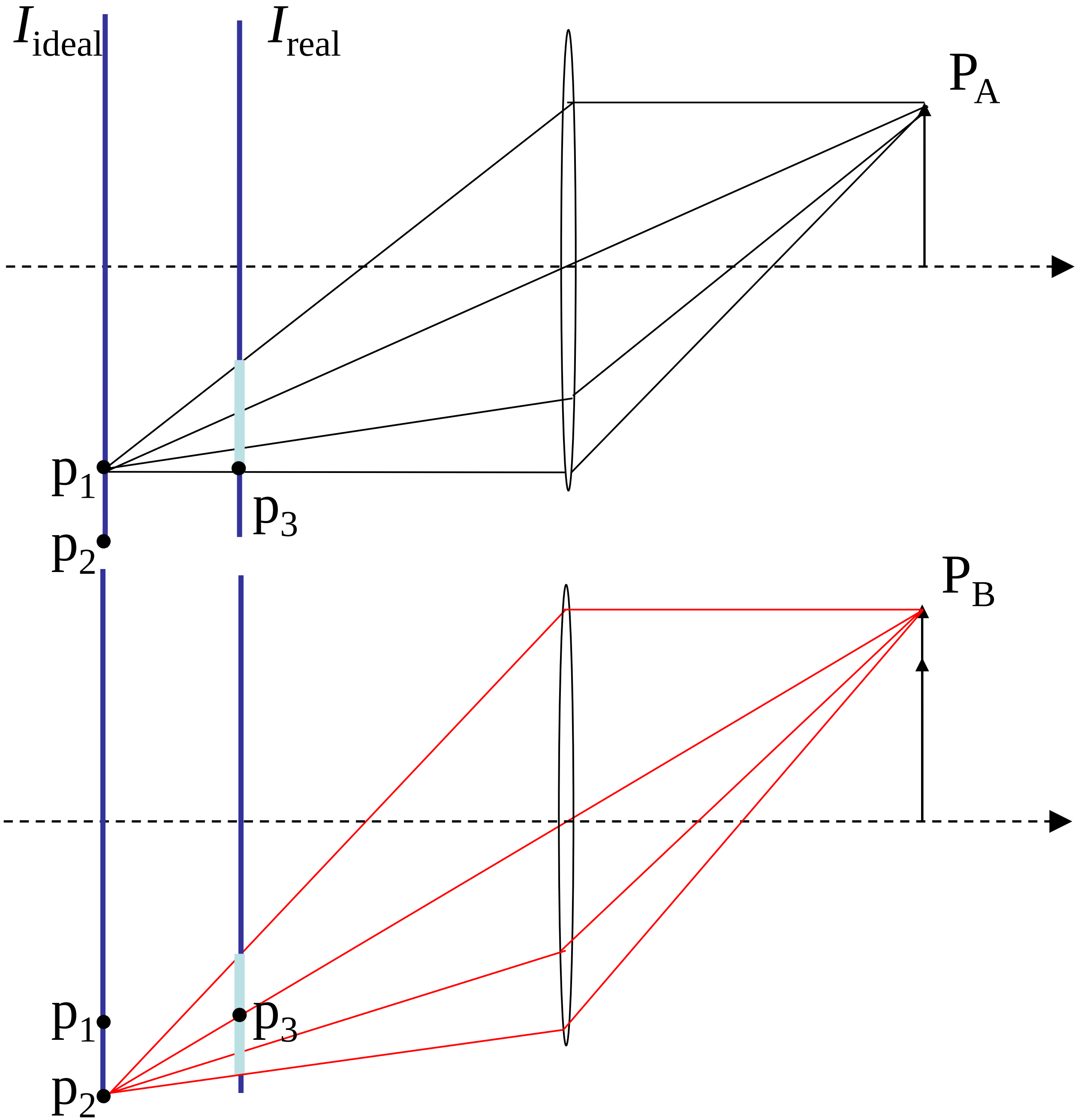
Expand/Pad



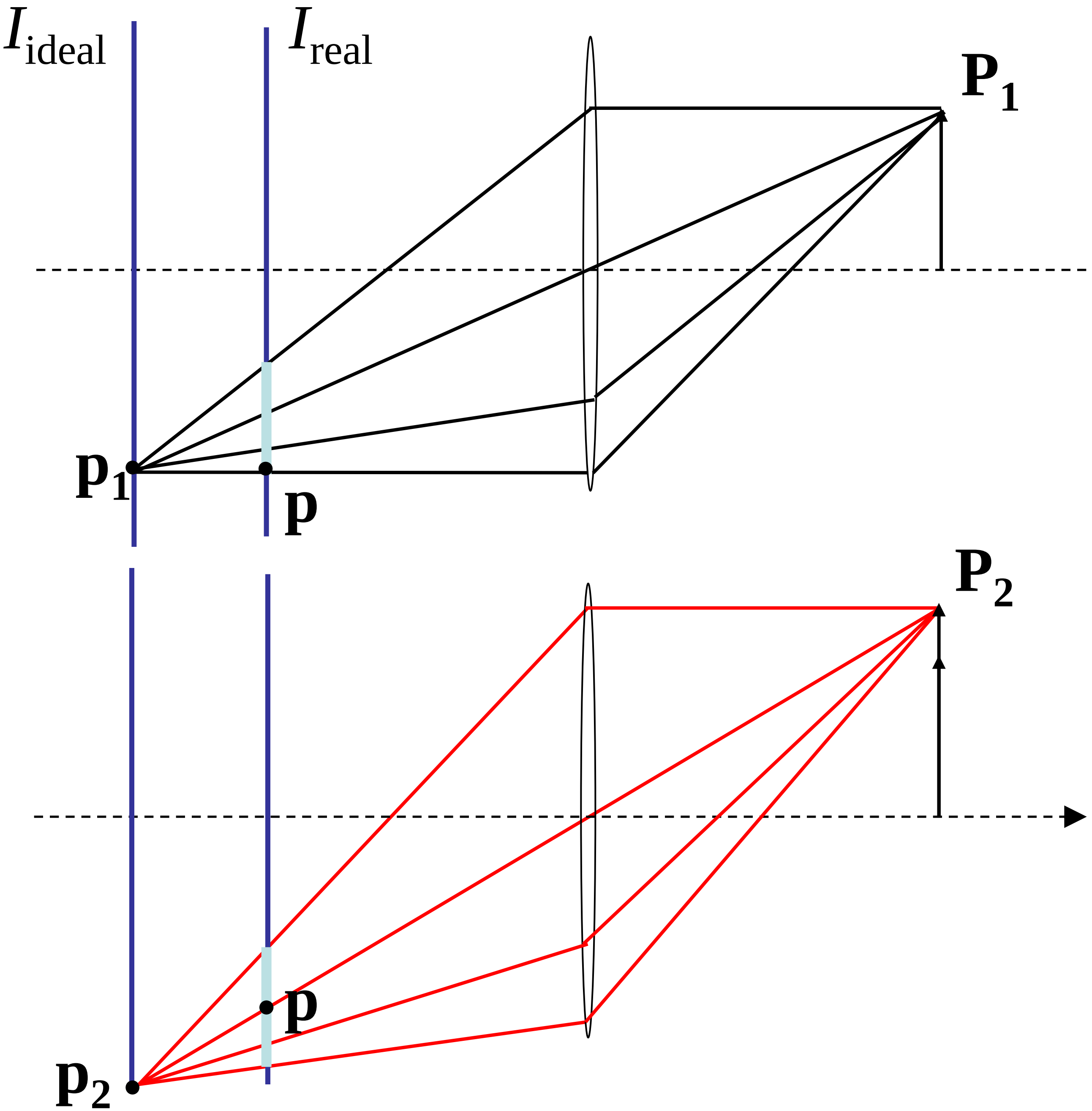
Crop



Defocus = Convolution?

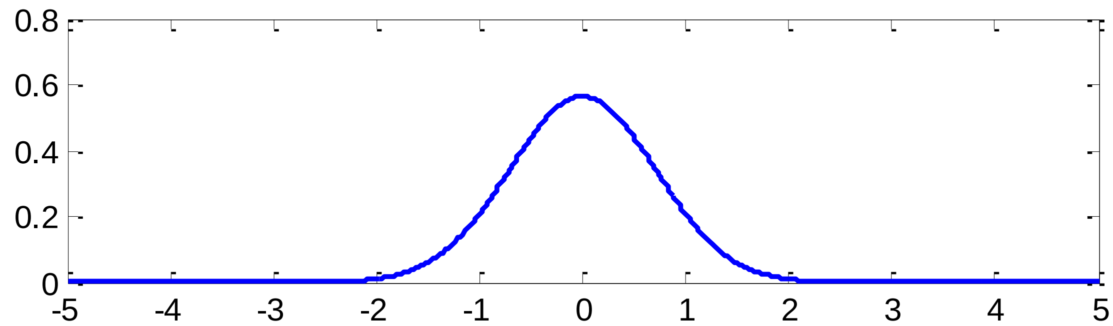


Defocus = Convolution?



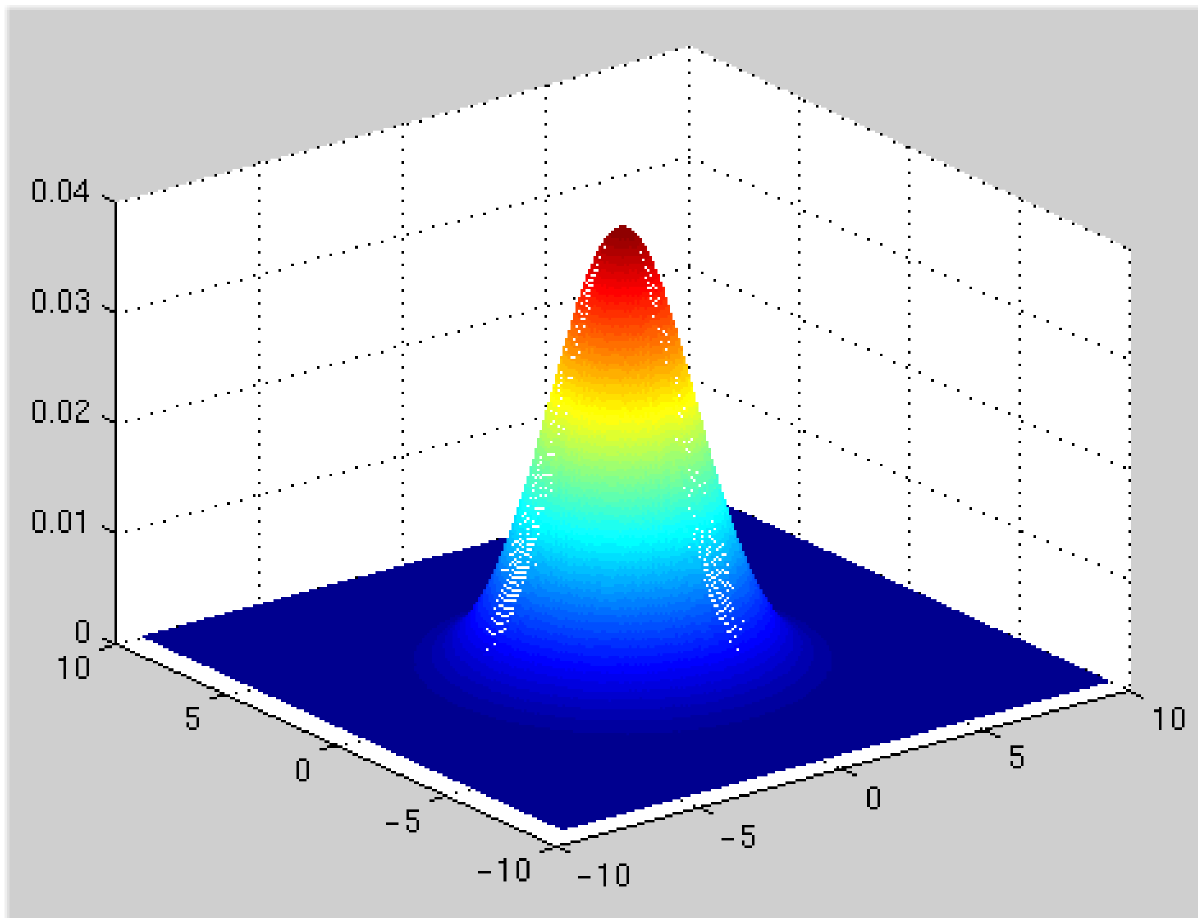
1-D:

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



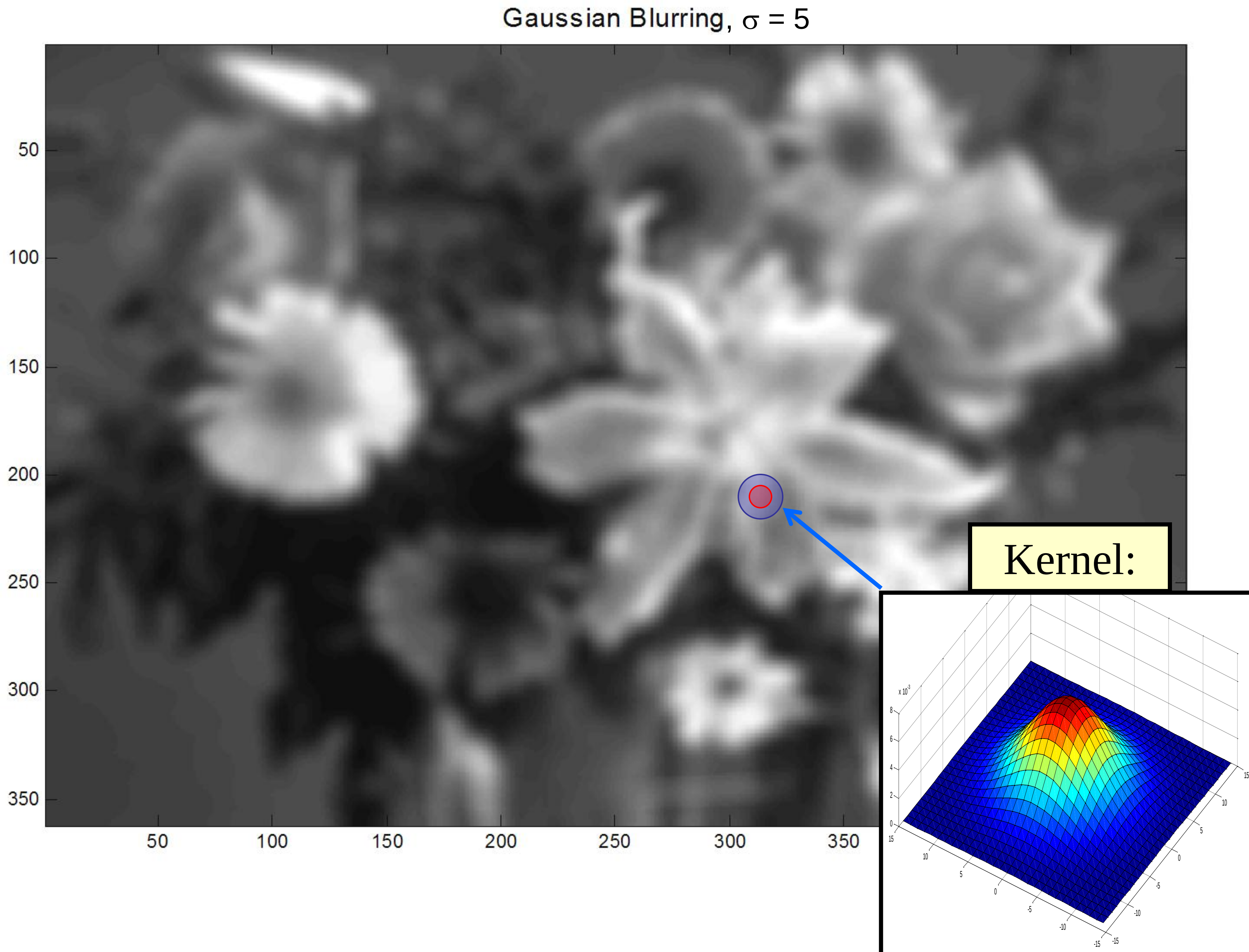
2-D:

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

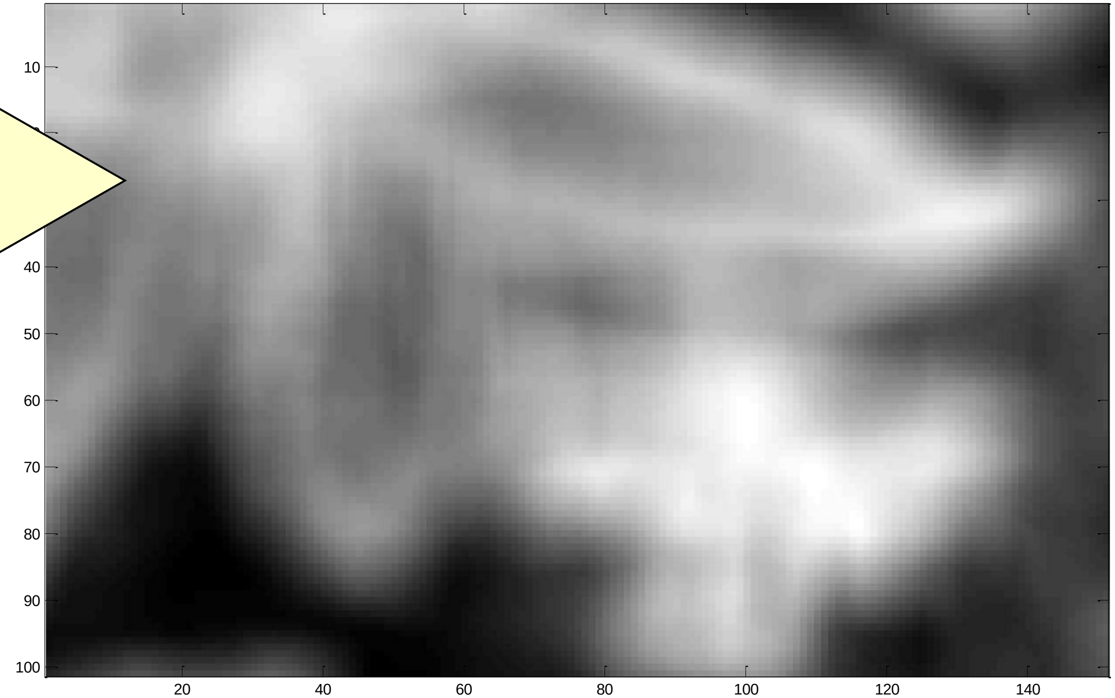


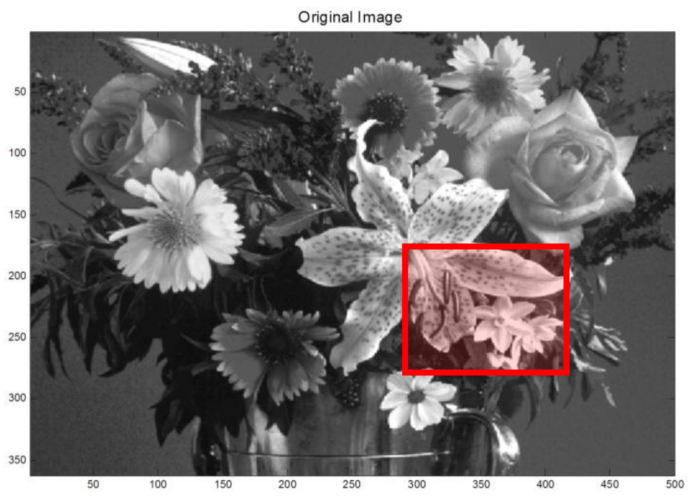
Slight abuse of notations:
We ignore the normalization constant such that

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


Gaussian Blurring, σ = 5
50
100
150
200
250
300
350
50
100
150
200
250
300
350
Kernel:

Simple Averaging





Gaussian Smoothing

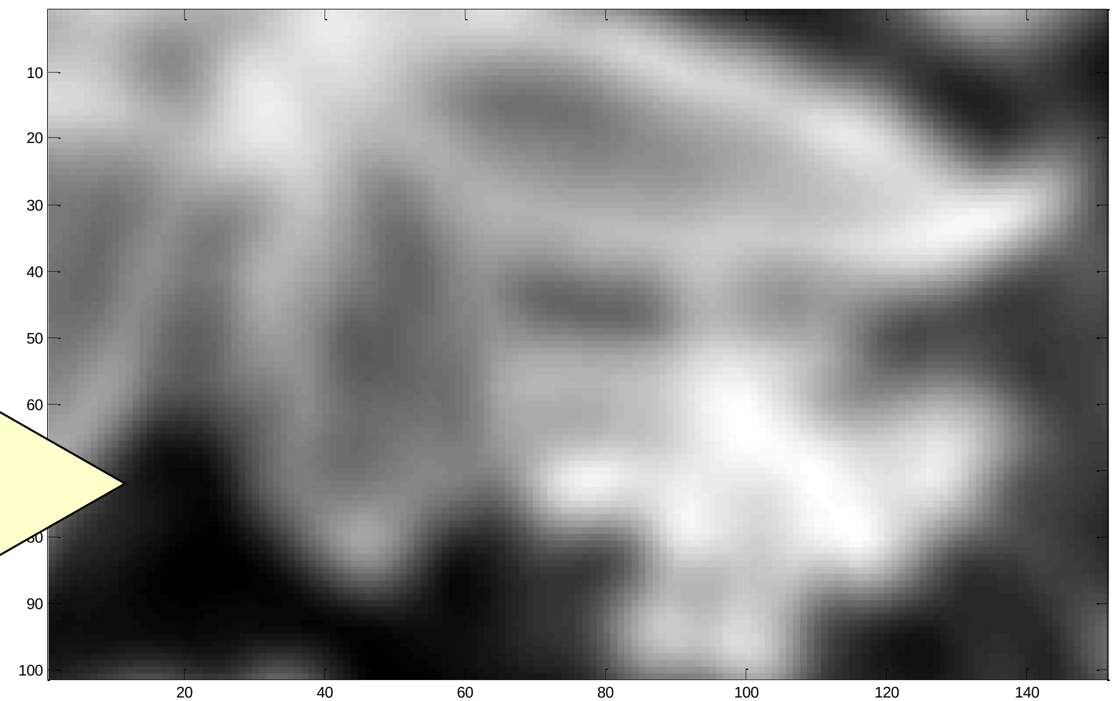
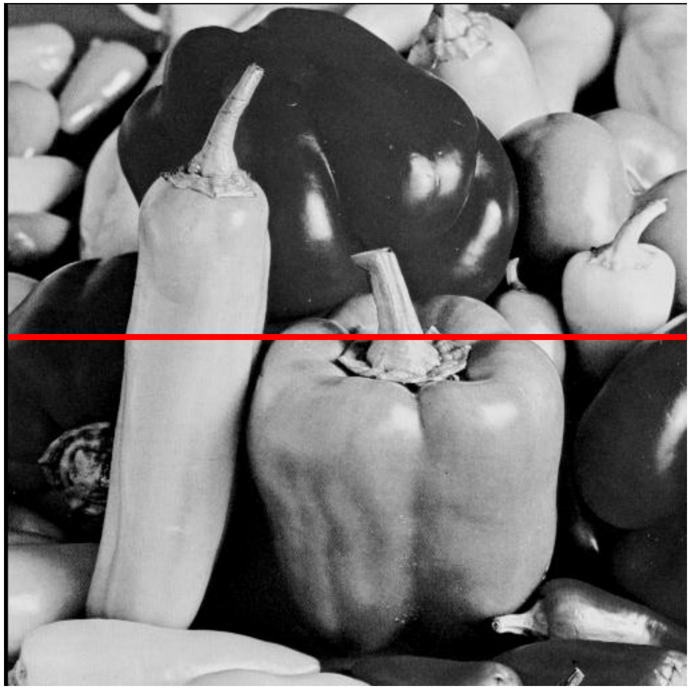
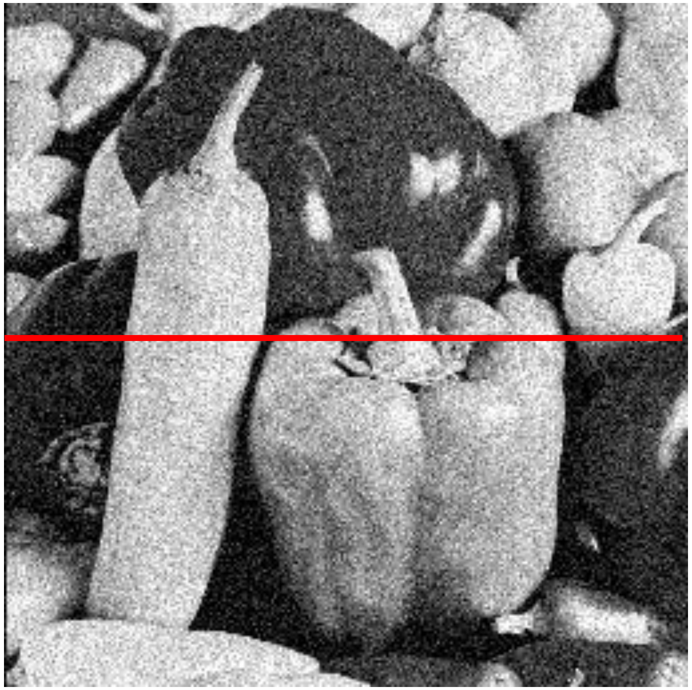
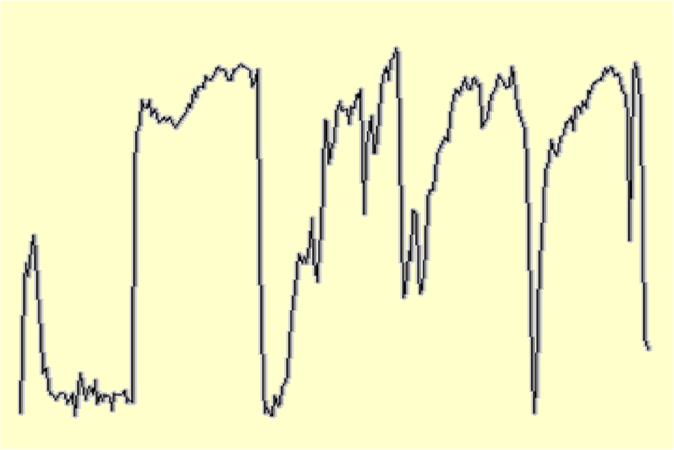
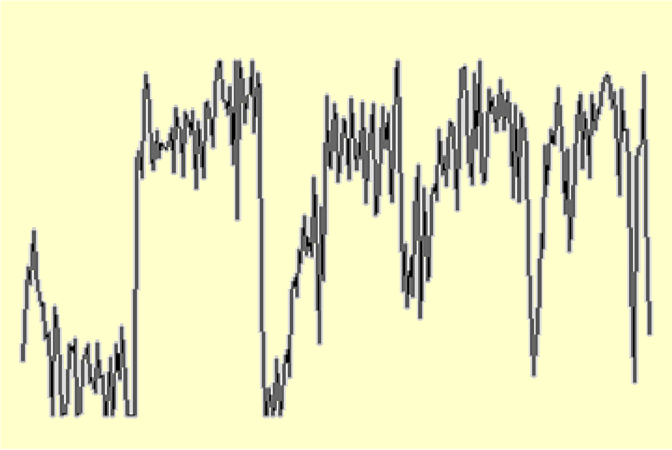


Image Noise





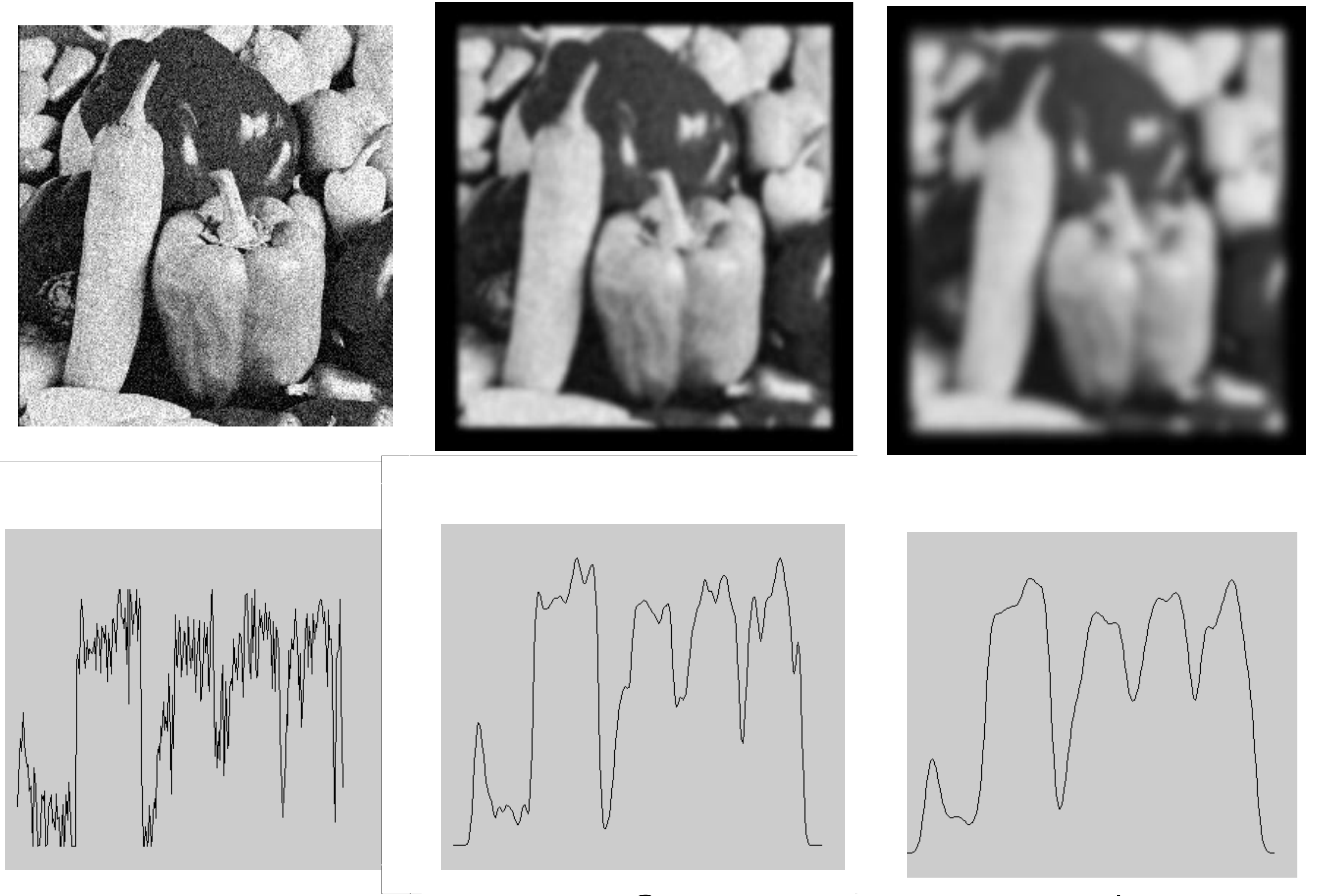




$$f(x,y) = \overbrace{\widehat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

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

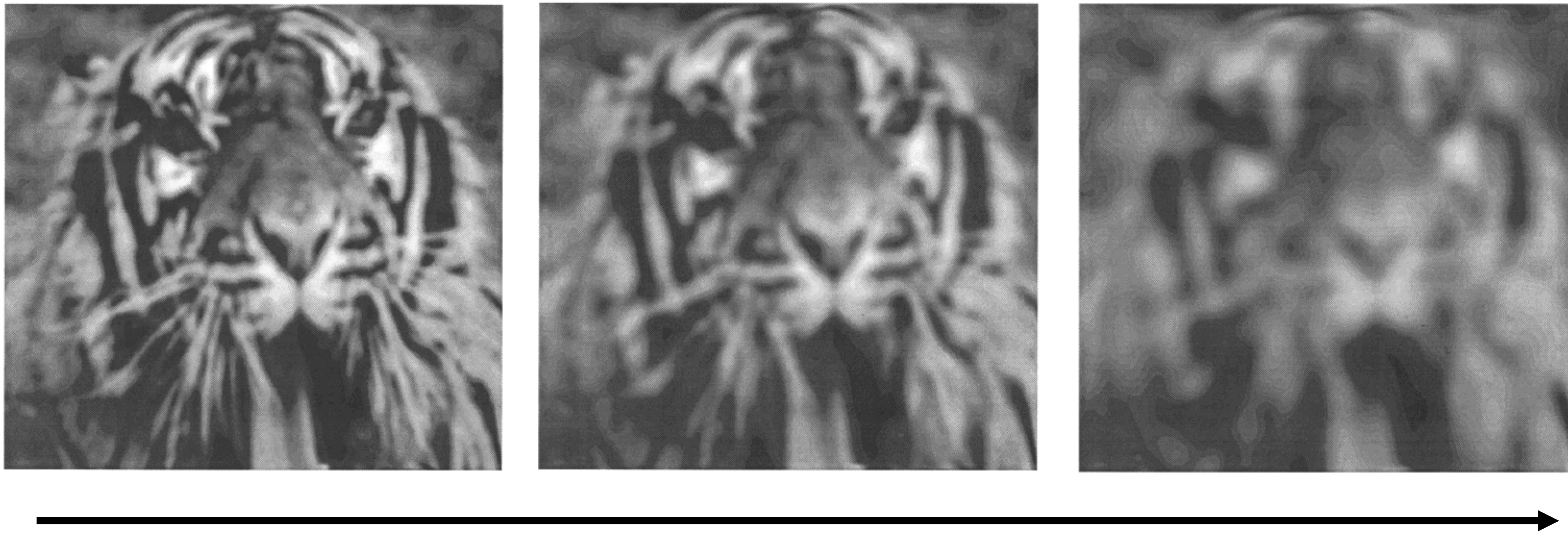
Gaussian Smoothing to Remove Noise



No smoothing

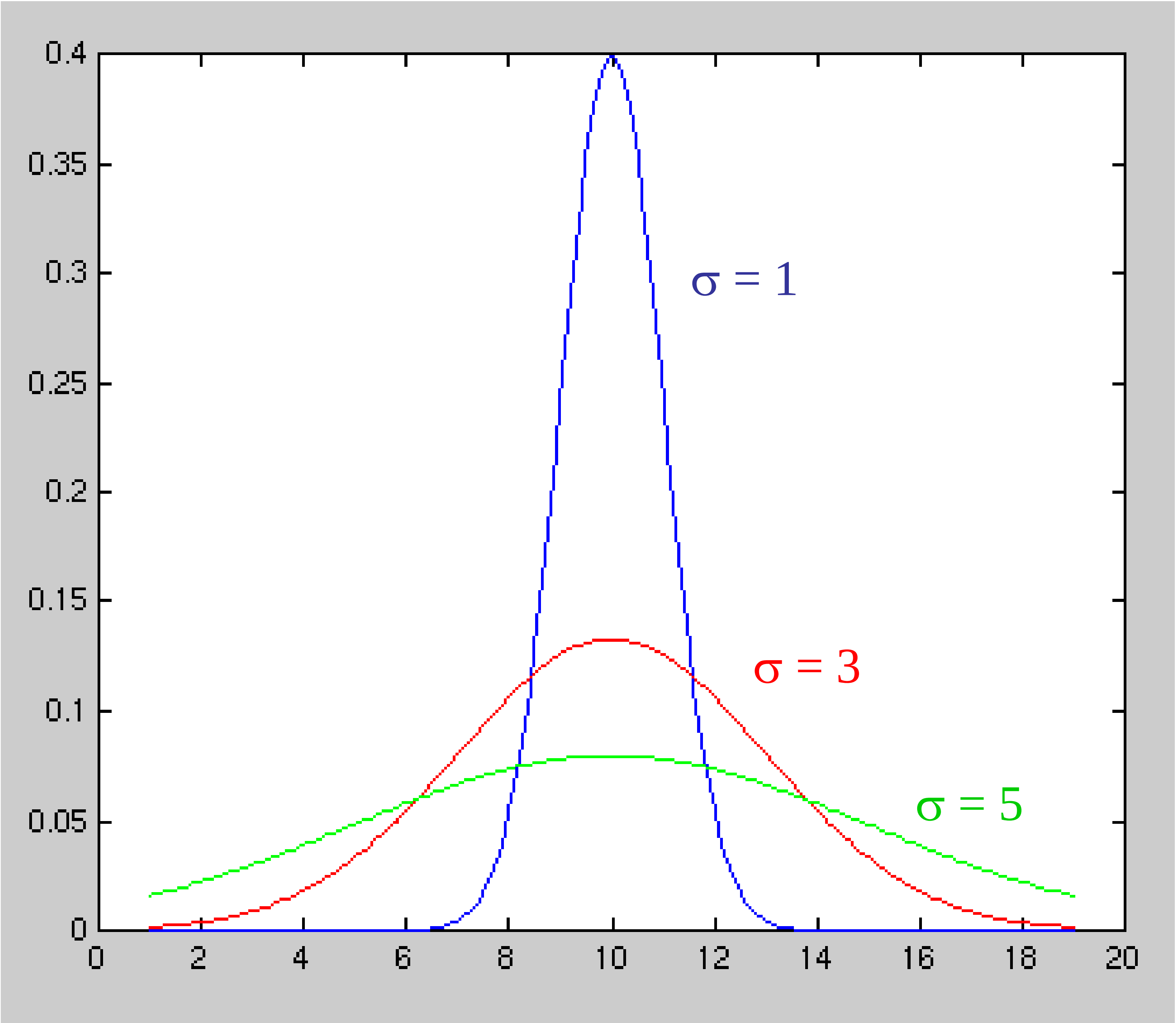
$\sigma = 2$

$\sigma = 4$



Increasing σ

Shape of Gaussian filter as function of σ



Basic Properties

- Gaussian removes "high-frequency" components from the image → "low pass" filter
- Larger σ remove more details
- Combination of 2 Gaussian filters is a Gaussian filter:

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

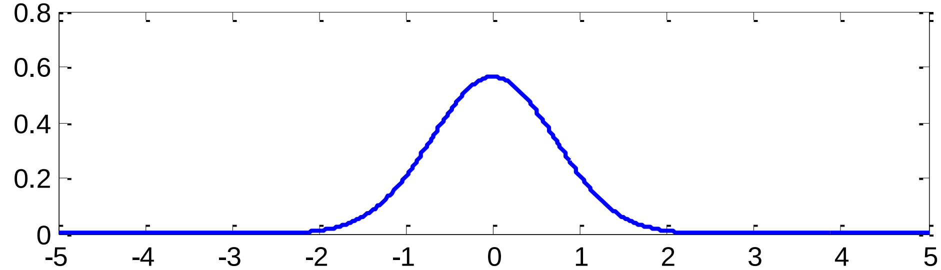
- Separable filter:

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

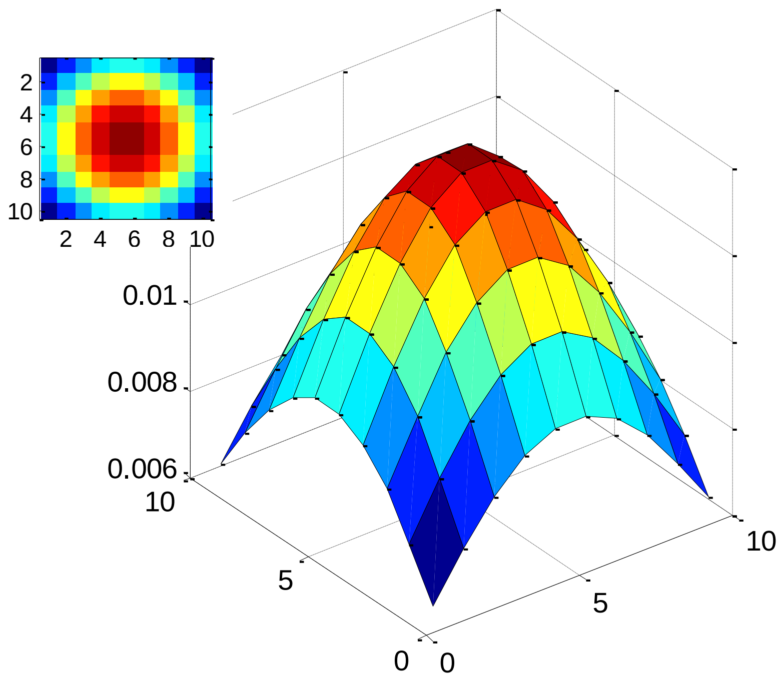
- Critical implication: Filtering with a *N*x*N* Gaussian kernel can be implemented as two convolutions of size *N* → reduction quadratic to linear → *must* be implemented that way

Note about Finite Kernel Support

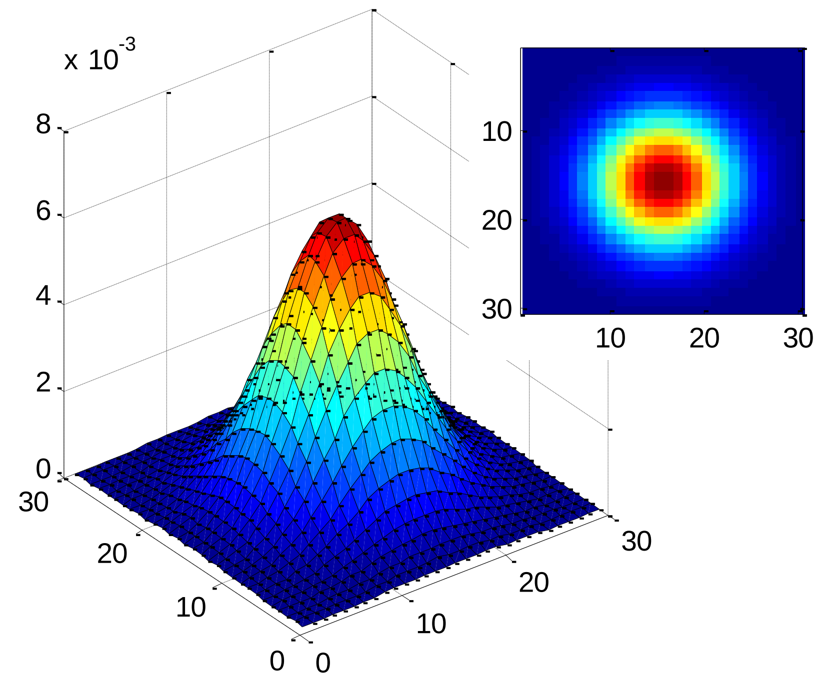
- Gaussian function has infinite support



- In discrete filtering, we have finite kernel size



σ = 5 with 10x10 kernel

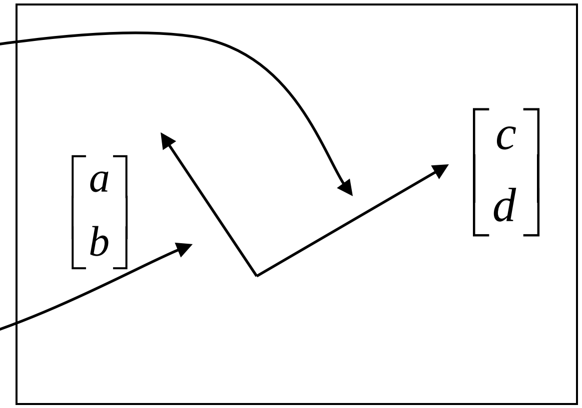


σ = 5 with 30x30 kernel

Oriented Gaussian Filters

- G_σ 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 σ_1 in the direction defined by the unit vector [a b] and by σ_2 in the direction defined by [c d]

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

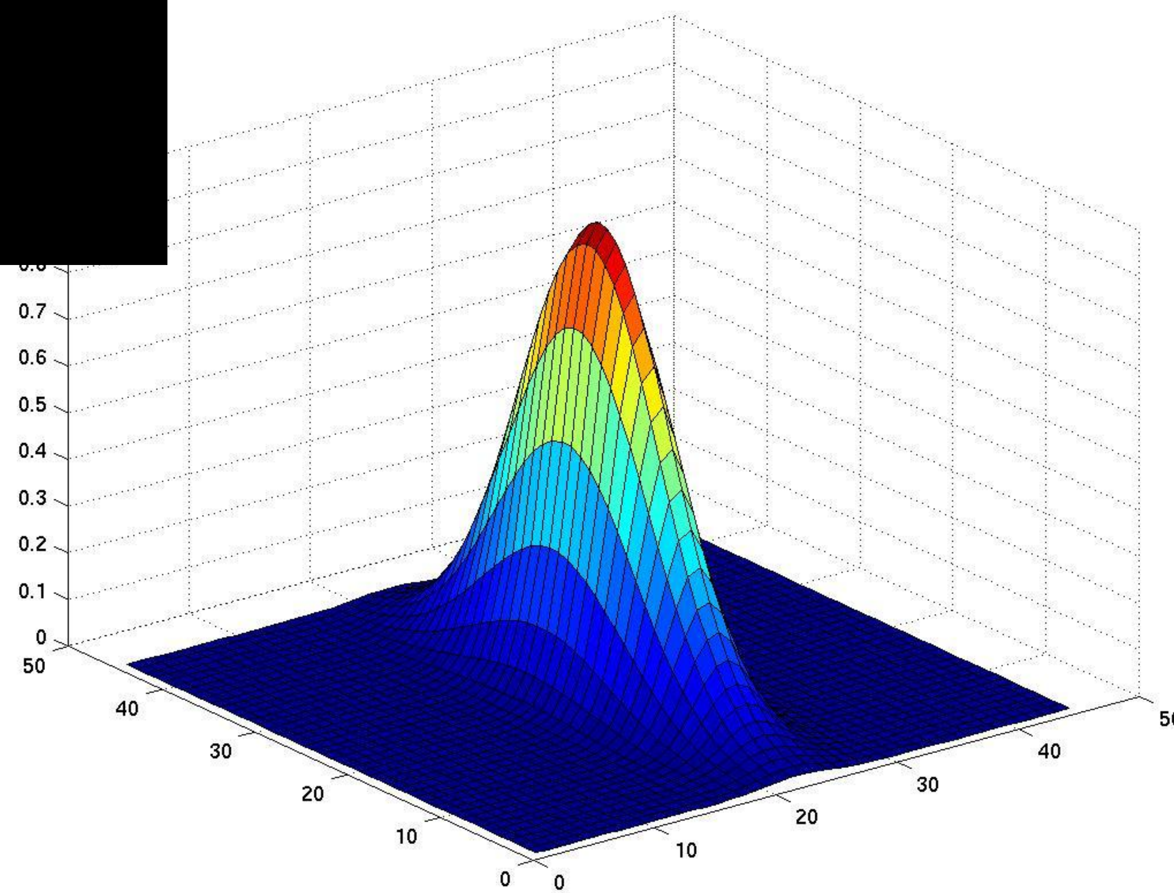


- We can write this in a more compact form by using the standard multivariate Gaussian notation:

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

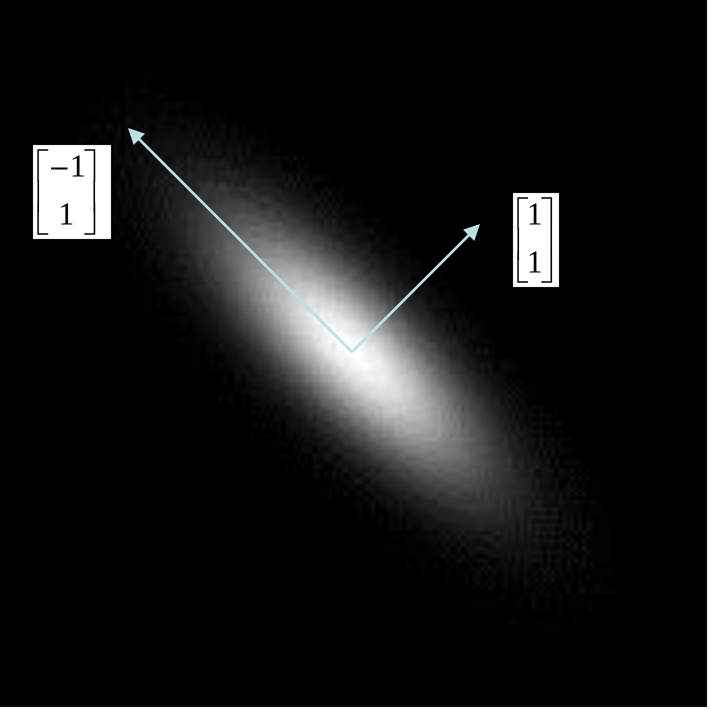
- The two (orthogonal) directions of filtering are given by the eigenvectors of Σ, the amount of smoothing is given by the square root of the corresponding eigenvalues of Σ.

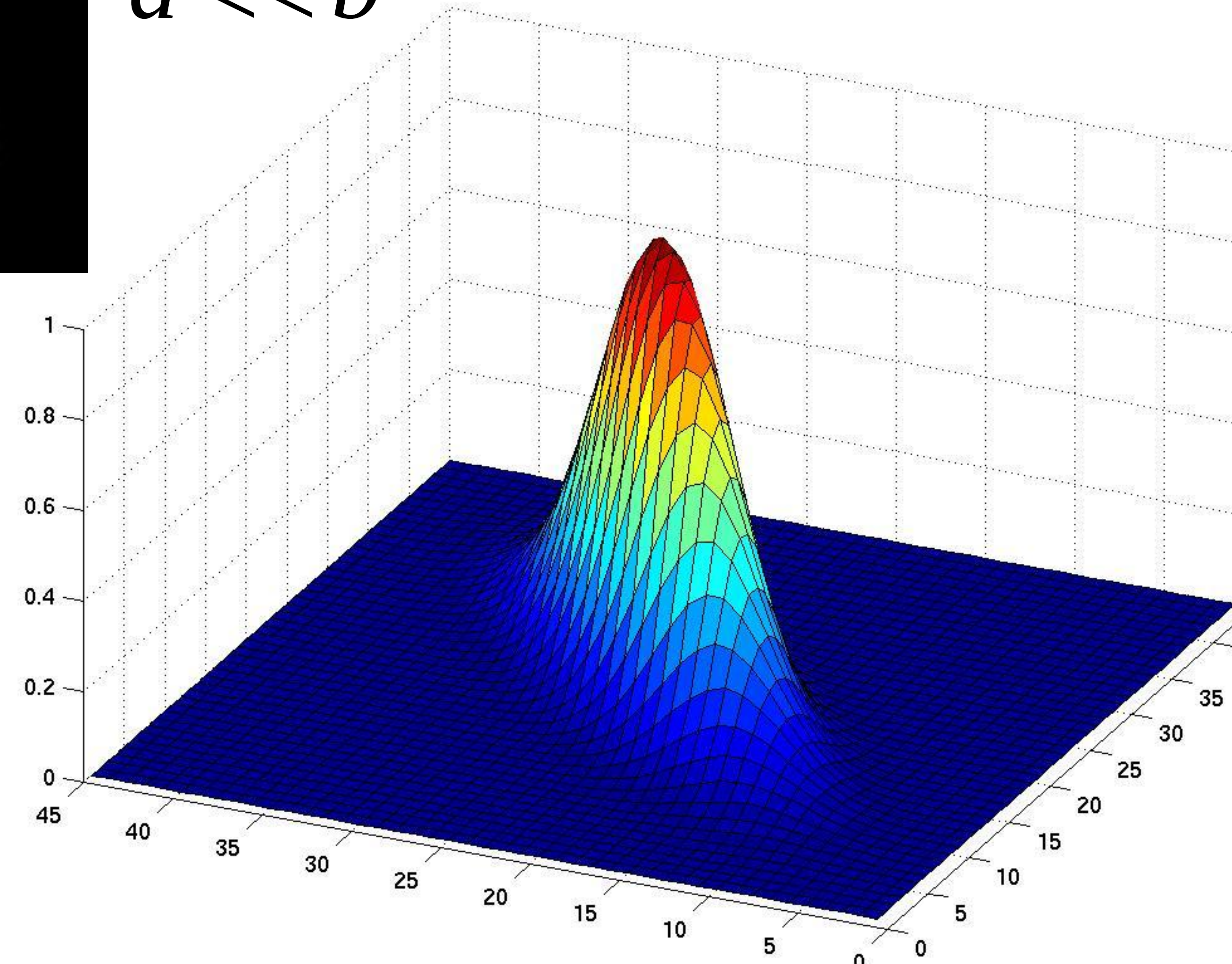
$$\Sigma = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad a << b$$



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

$$a << b$$





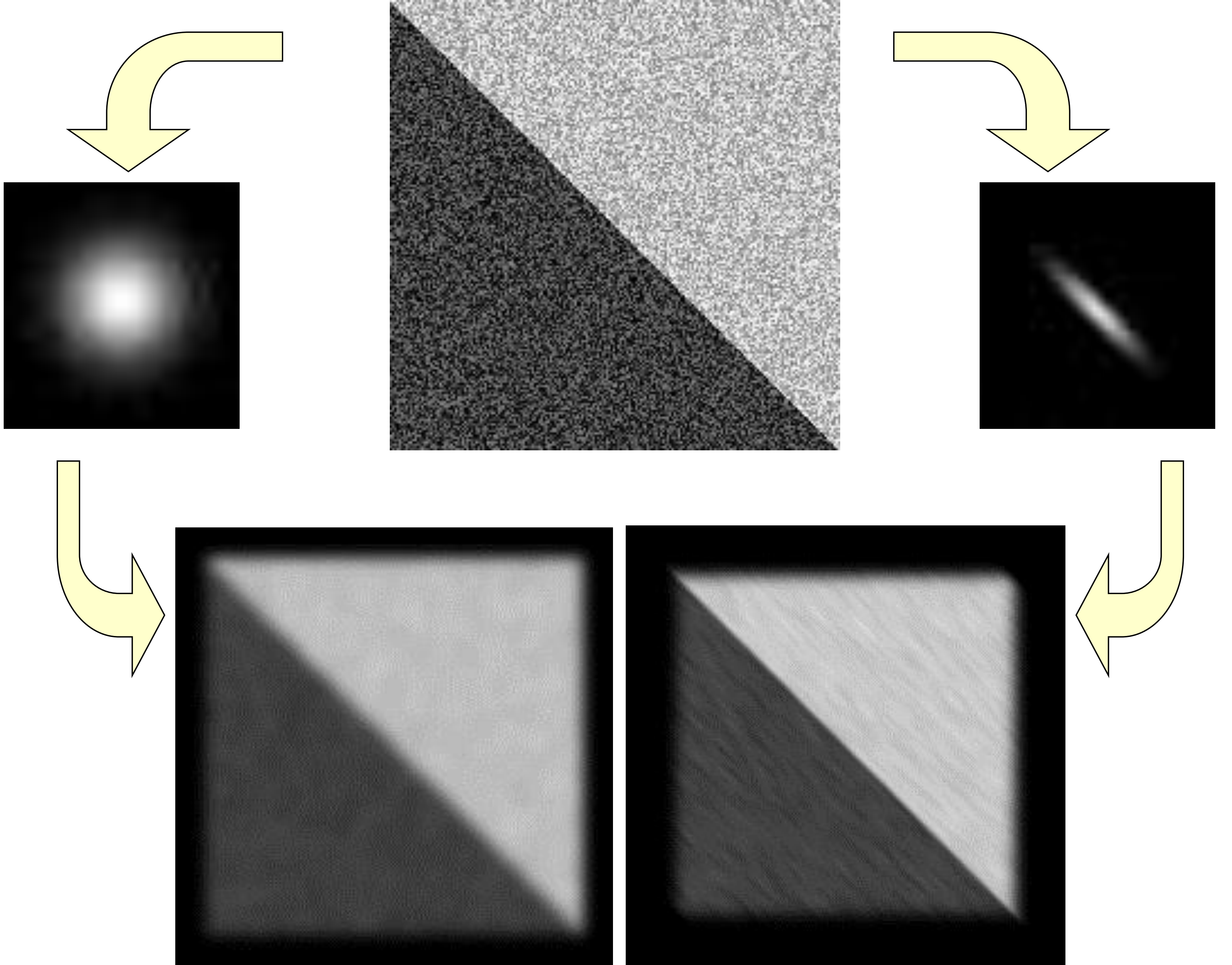


Image Derivatives

- 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

$$\partial_x = \boxed{-1 \mid 0 \mid 1}$$

- Problem: Increases noise

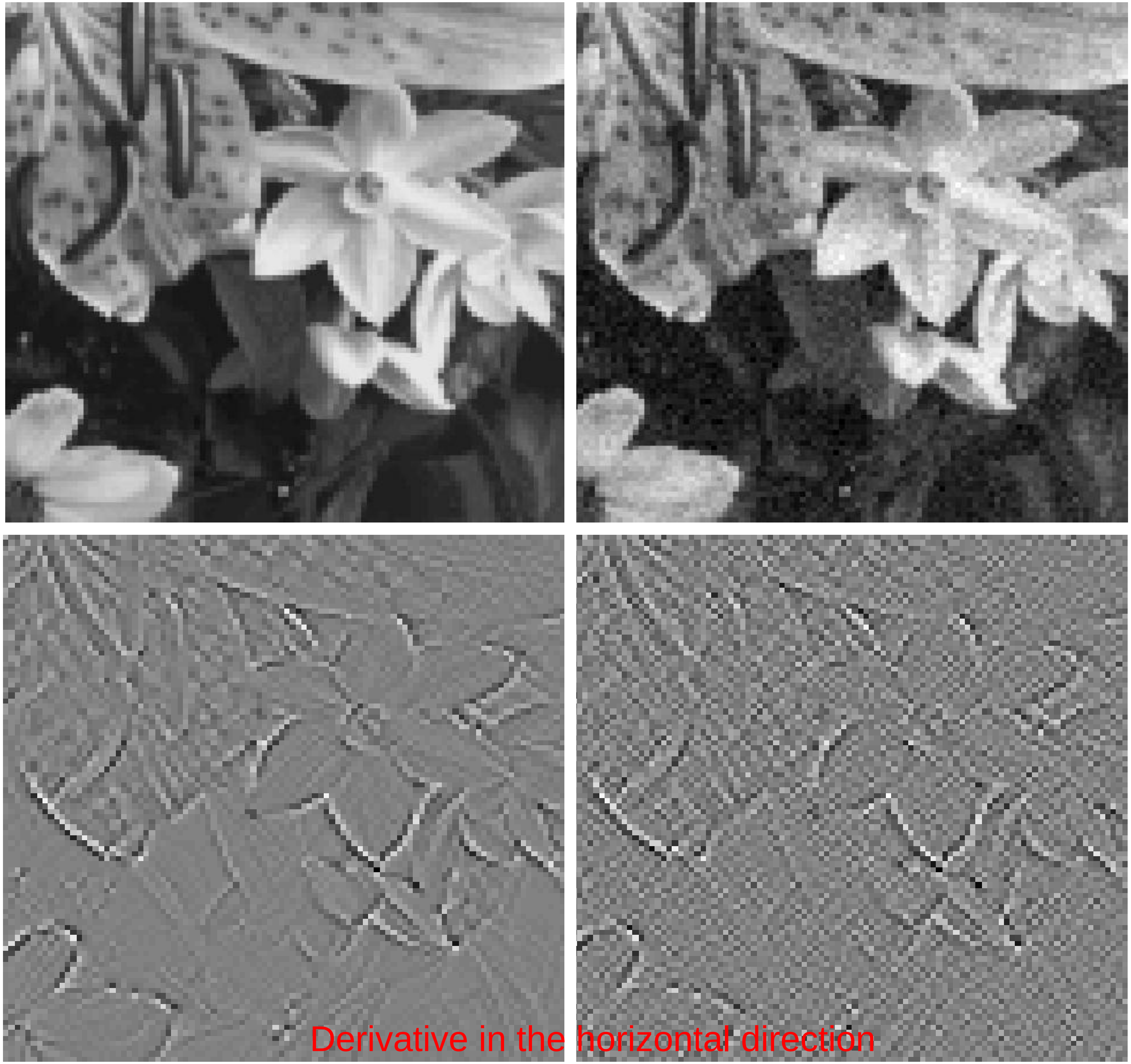
$$\underbrace{I(i+1, j) - I(i-1, j)}_{\text{Difference between Actual image values}} = \underbrace{\hat{I}(i+1, j) - \hat{I}(i-1, j)}_{\text{True difference (derivative)}} \underbrace{+ n_+ + n_-}_{\text{Twice the amount of noise as in the original image}}$$

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_σ and then take derivatives:

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

- Applying the differentiation property of the convolution:

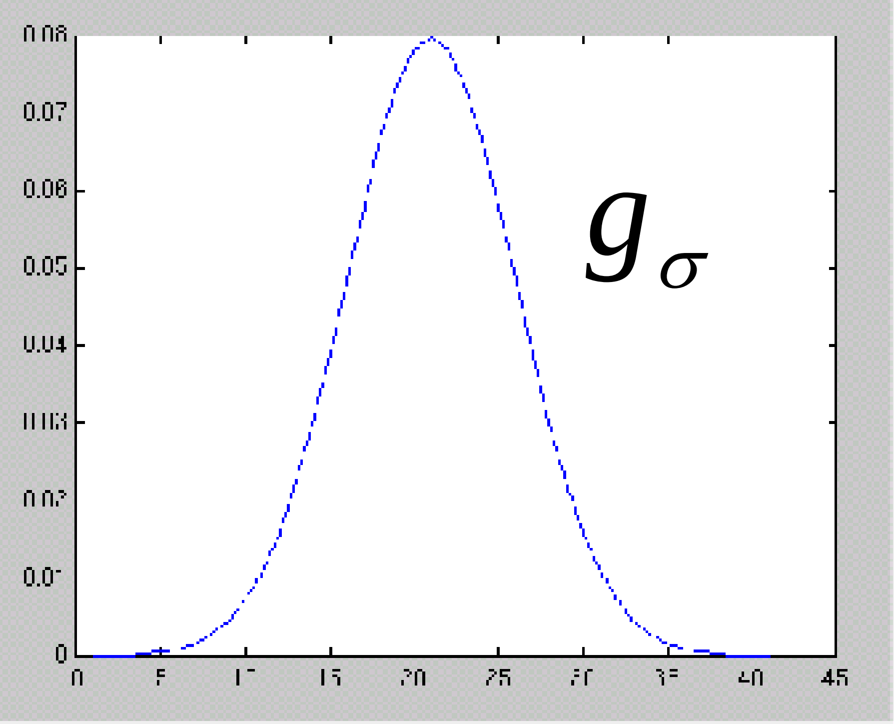
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

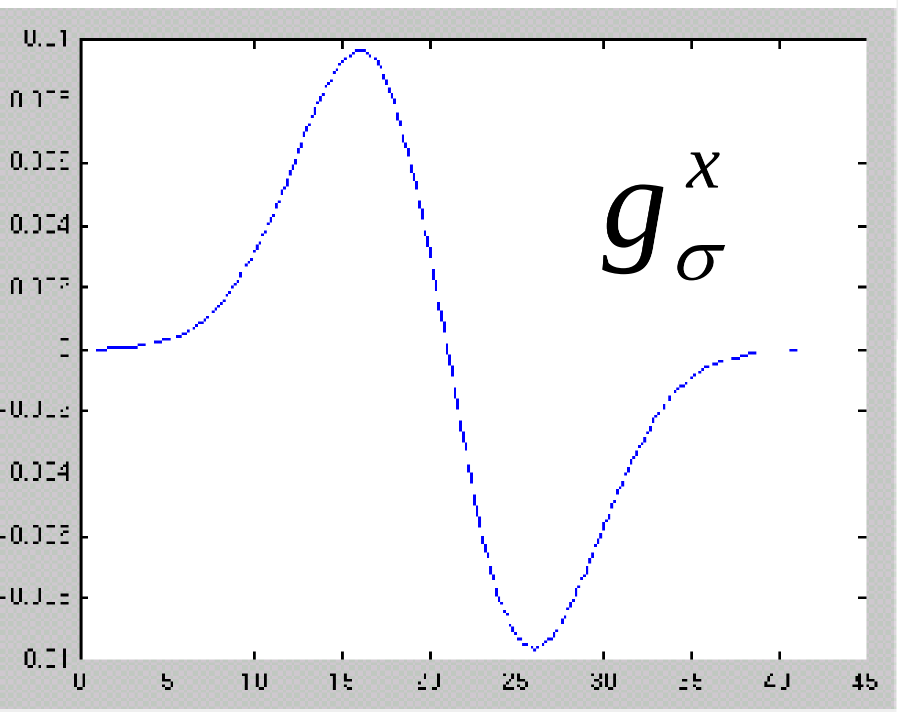
- Therefore, taking the derivative in x of the image can be done by convolution with the derivative of a Gaussian:

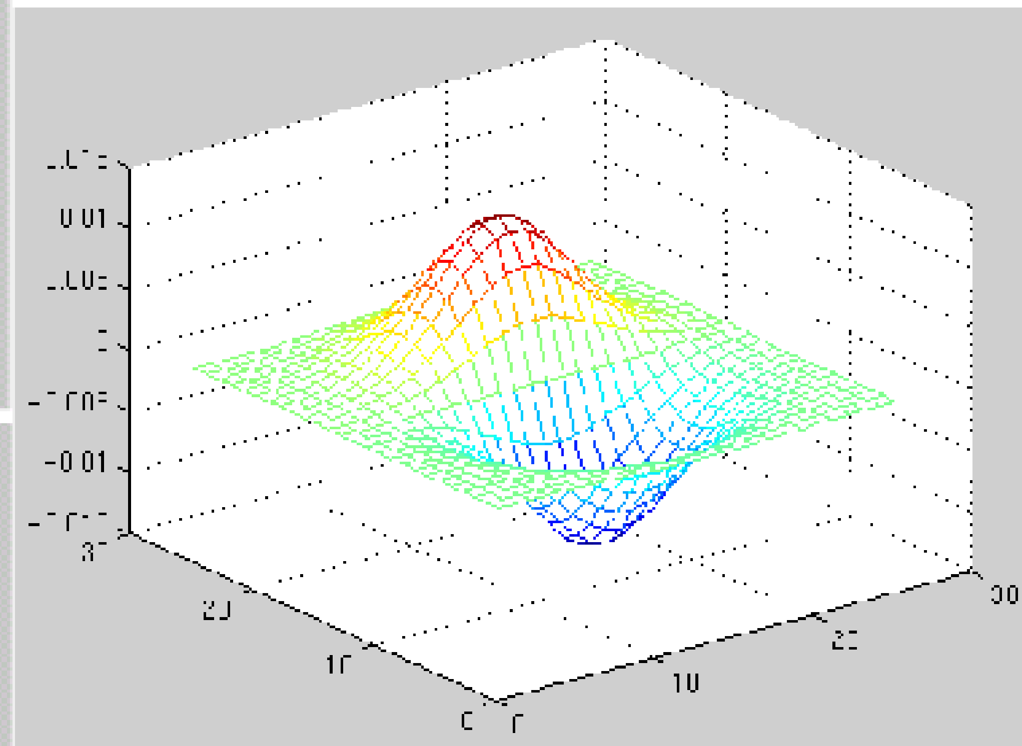
$$G_\sigma^x = \frac{\partial G_\sigma}{\partial x} = xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Crucial property: The Gaussian derivative is also separable:

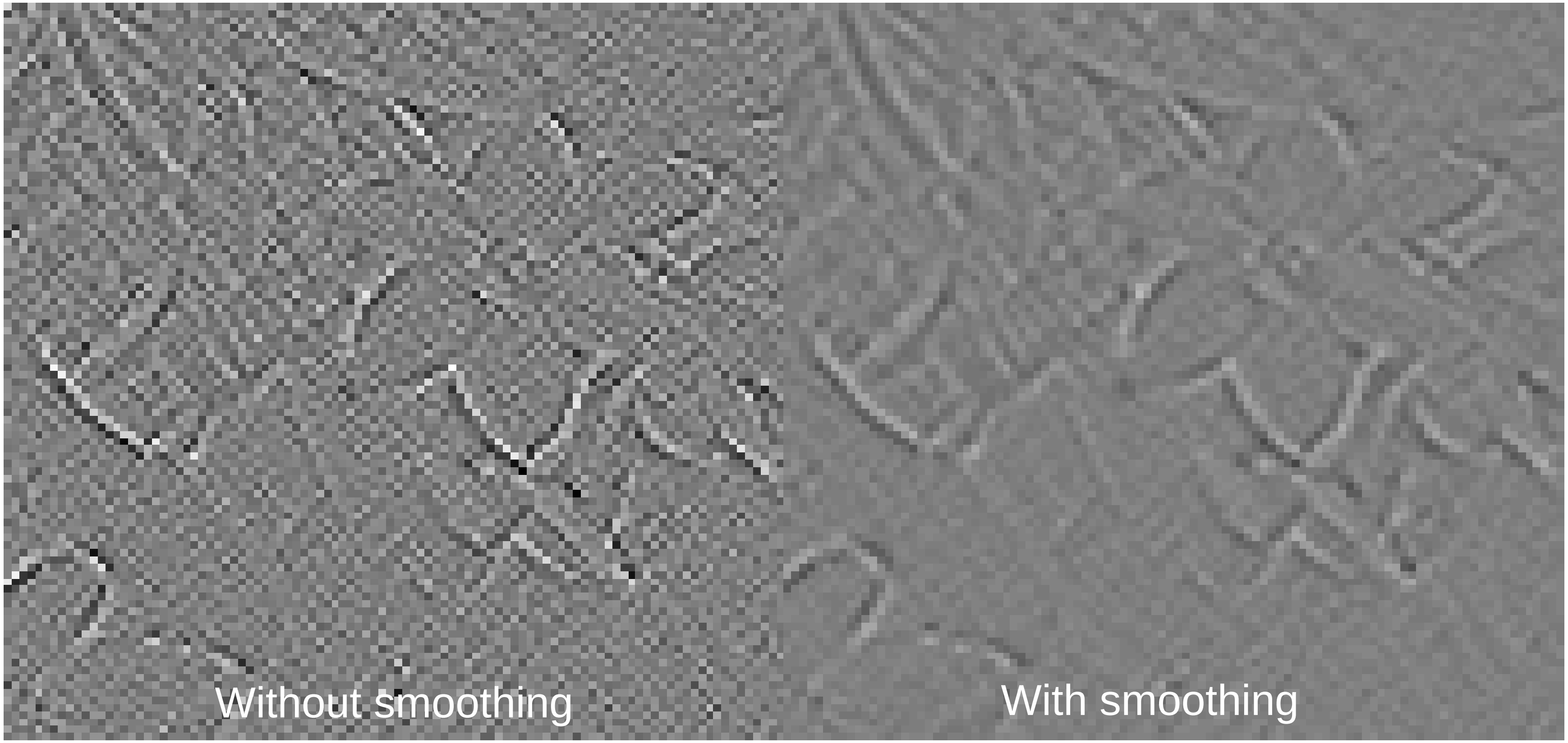
$$G_\sigma^x * f = g_\sigma^x * g_{\sigma\uparrow} * f$$


g_σ
0.08
0.07
0.06
0.05
0
0
5
15
20
25
40
45


g_σ^x
0
5
10
15
25
30
45


0.01
-0.01
10

Derivative + Smoothing



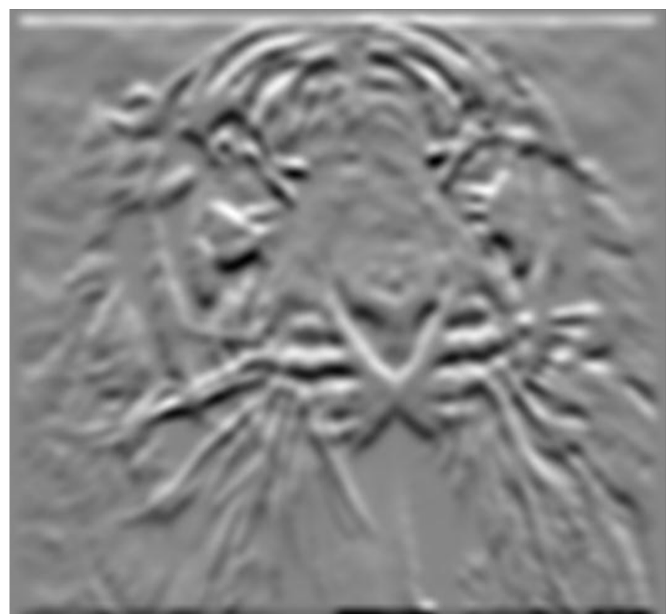
Better but still blurs away edge information

Applying the first derivative of Gaussian

I

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

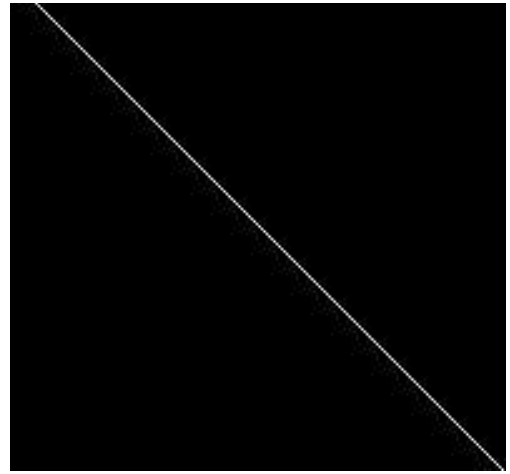
$$\frac{\partial I}{\partial x}$$



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

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

Image with Edge



Edge Location

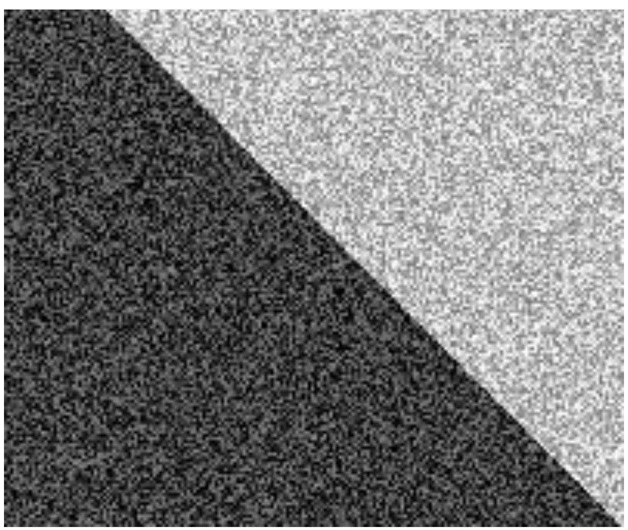
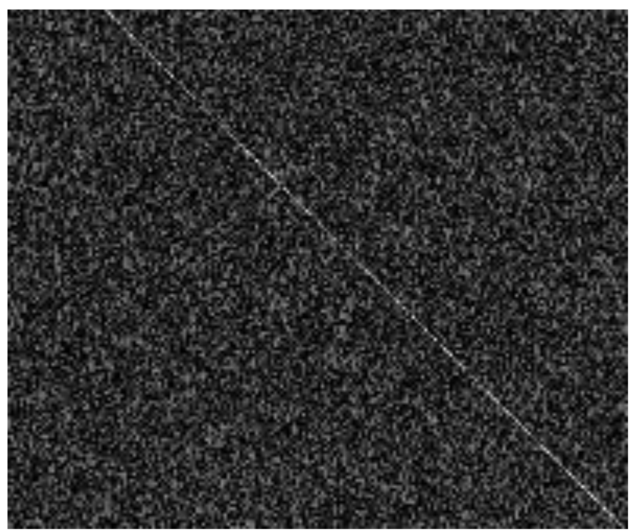
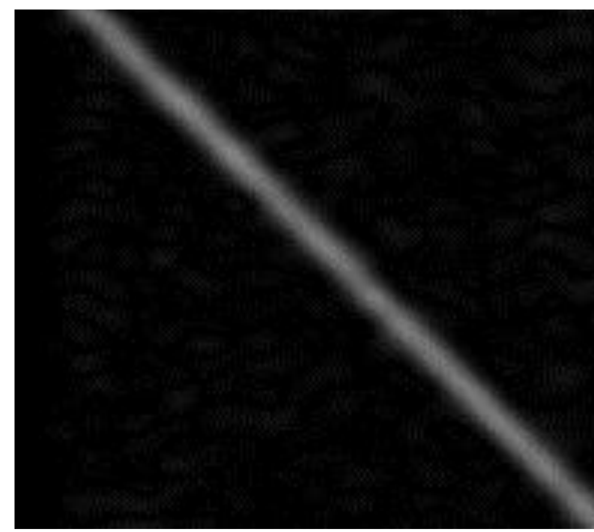


Image + Noise



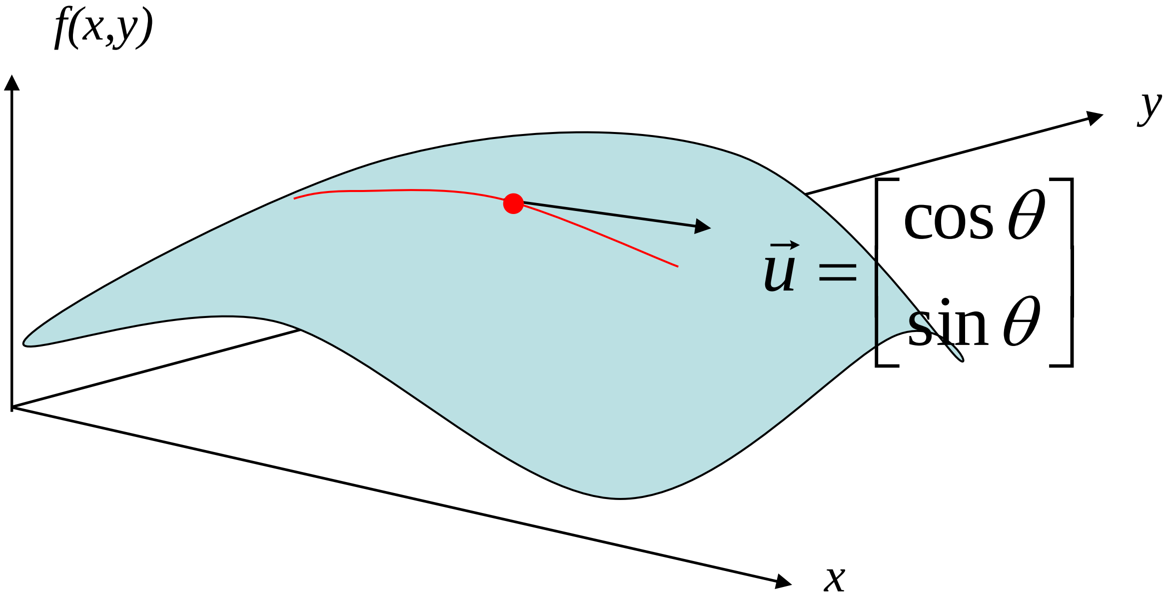
Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

Directional Derivatives

- In some cases, it might be interesting to compute the derivative of the image in some arbitrary direction defined by a unit vector:



- Conveniently, the derivative is obtained by convolution of the image with a simple linear combination of the axis derivatives:

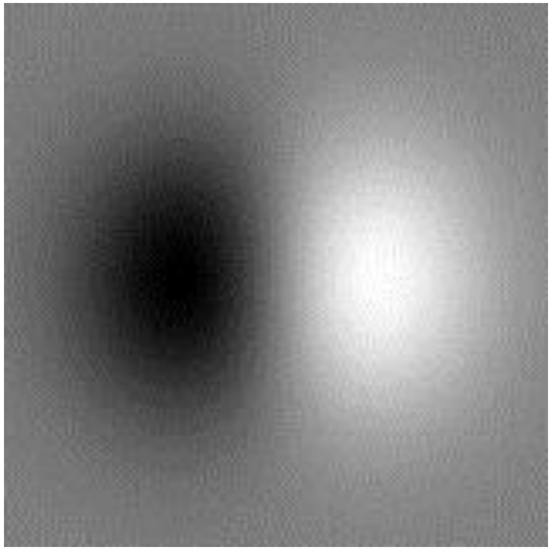
$$\frac{\partial f}{\partial \vec{u}} = (\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}) * f$$

- Note: More generally, filters that can be computed at any orientation as a linear combination of other, fixed filters are called steerable filters

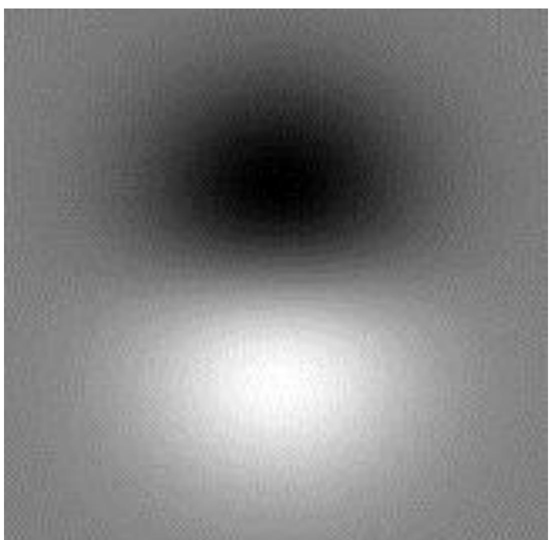
$$F(\theta) = \sum a_i(\theta) F_i$$

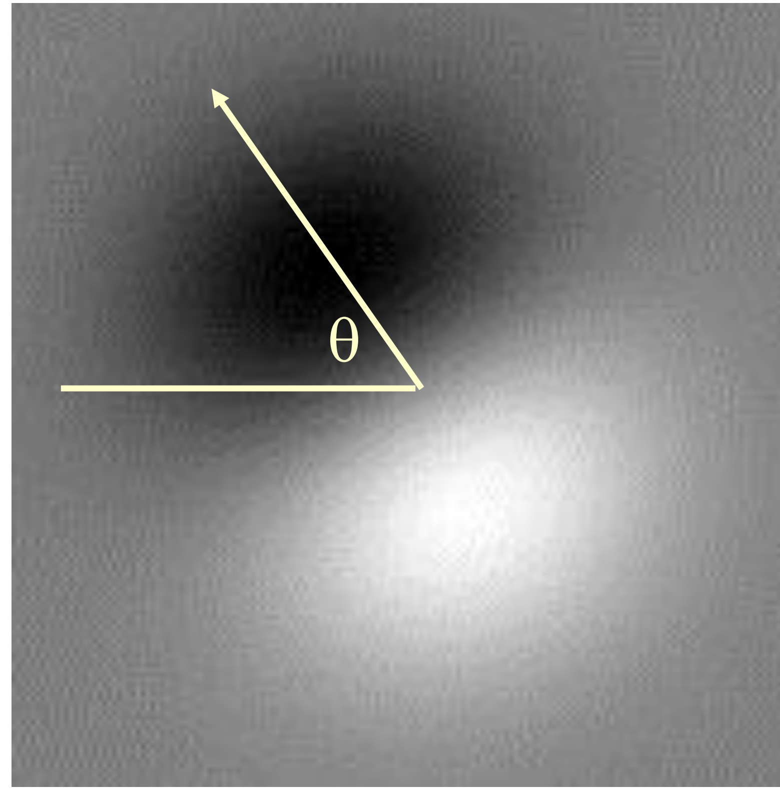
Directional Derivatives

$$\frac{\partial G_\sigma}{\partial x}$$



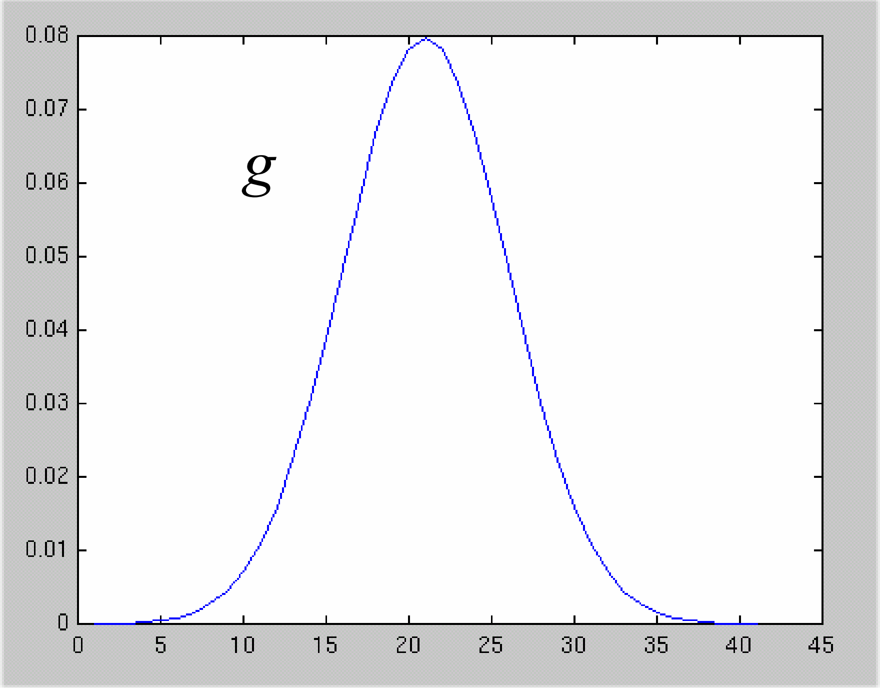
$$\frac{\partial G_\sigma}{\partial y}$$

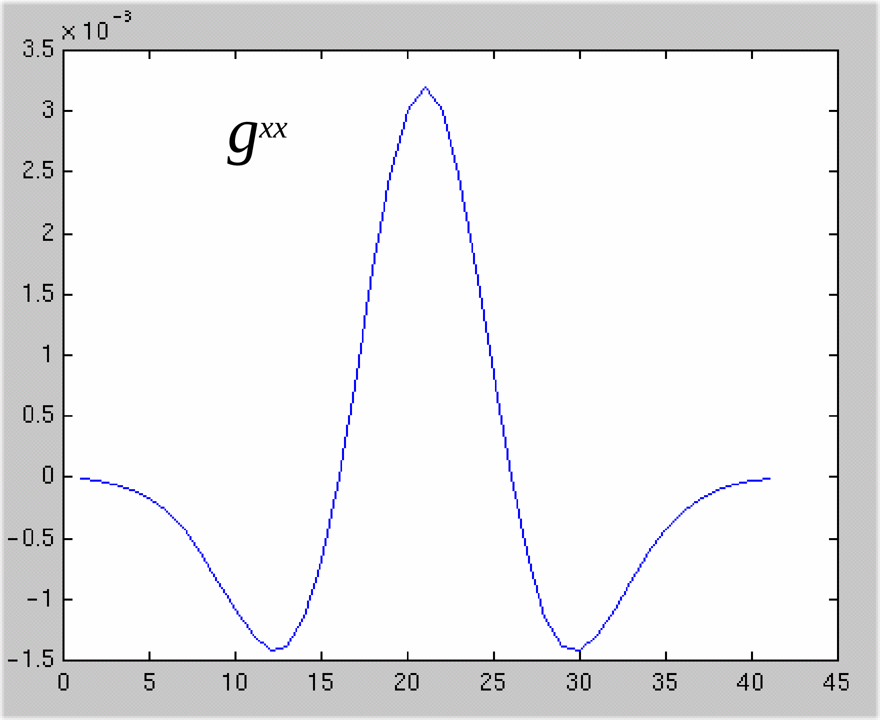


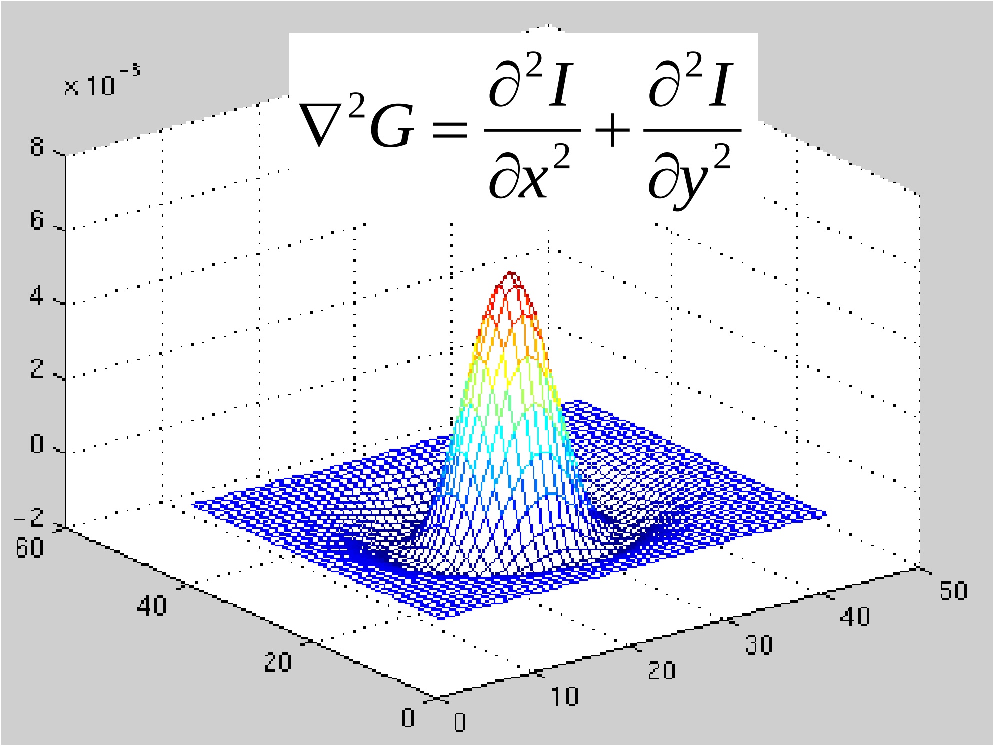


$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

Second derivatives: Laplacian



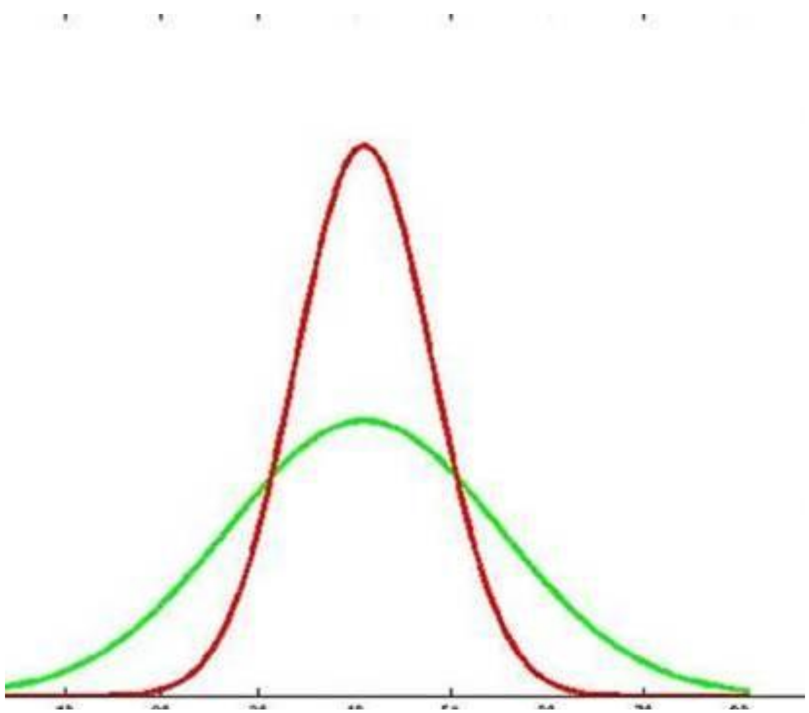


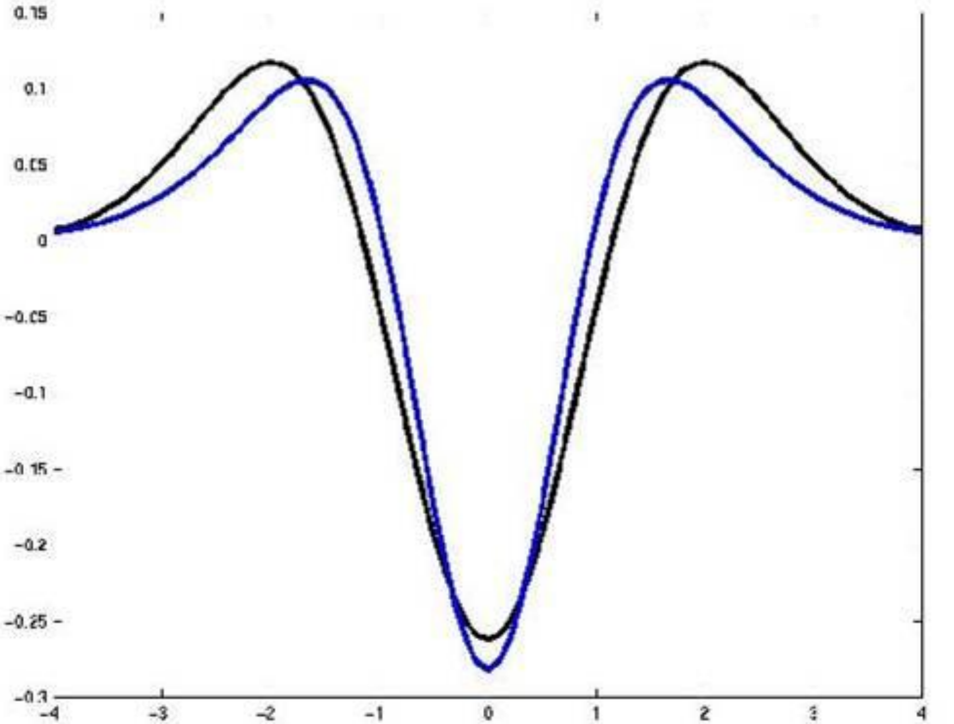


$$\nabla^2 G = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

DOG Approximation to LOG

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





Gaussian

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

Separable, low-pass filter

Derivatives of Gaussian

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

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

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

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 θ. Filter is linear combination of derivatives in x and y

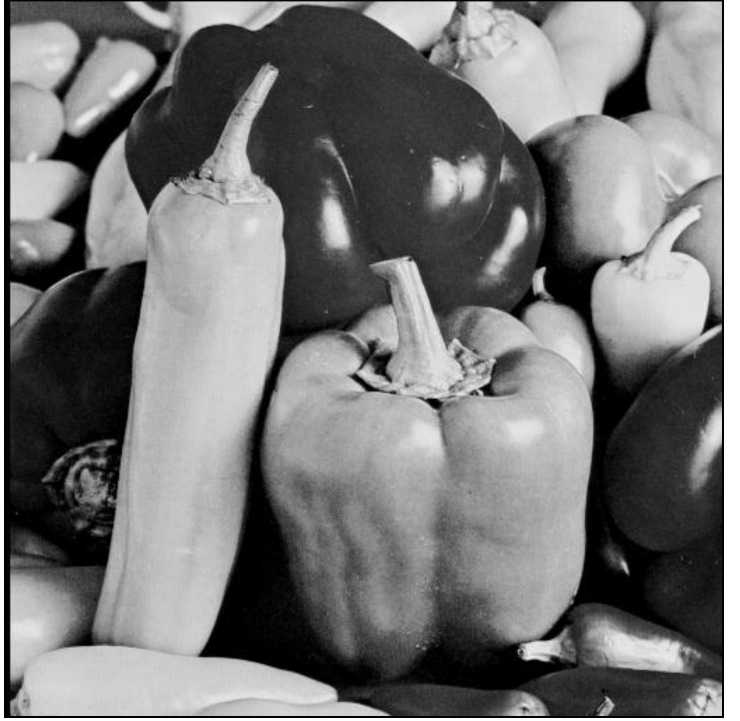
Oriented Gaussian

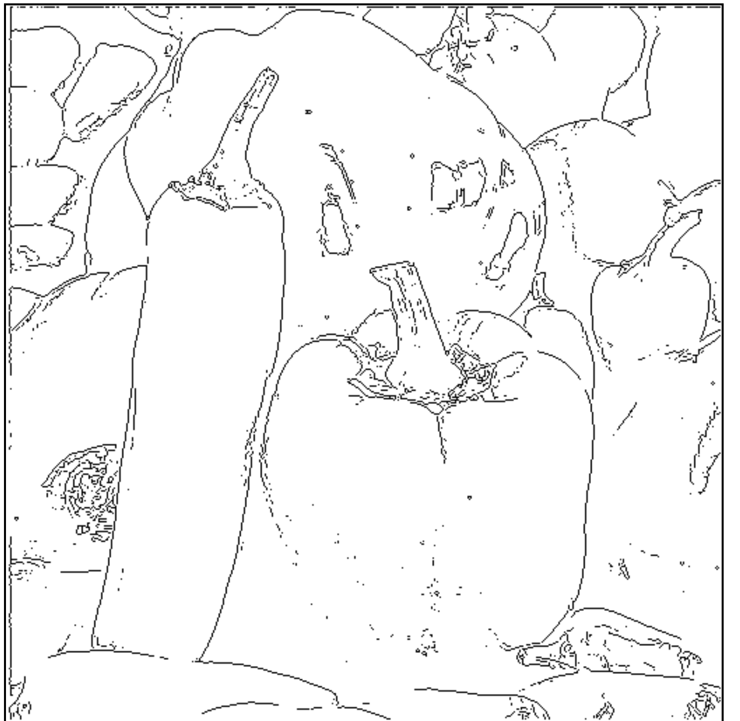
$$e^{-\frac{(a_1 x + b_1 y)^2}{2\sigma_1^2} - \frac{(a_2 x + b_2 y)^2}{2\sigma_2^2}}$$

Smooth with different scales in orthogonal directions

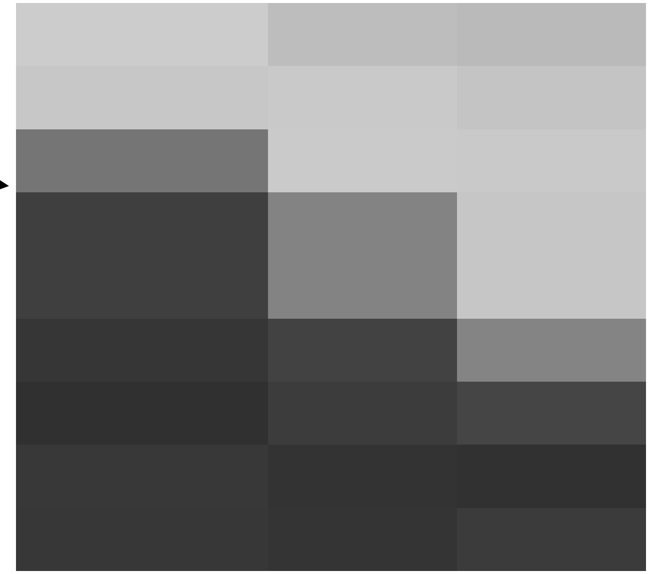
Edge Detection

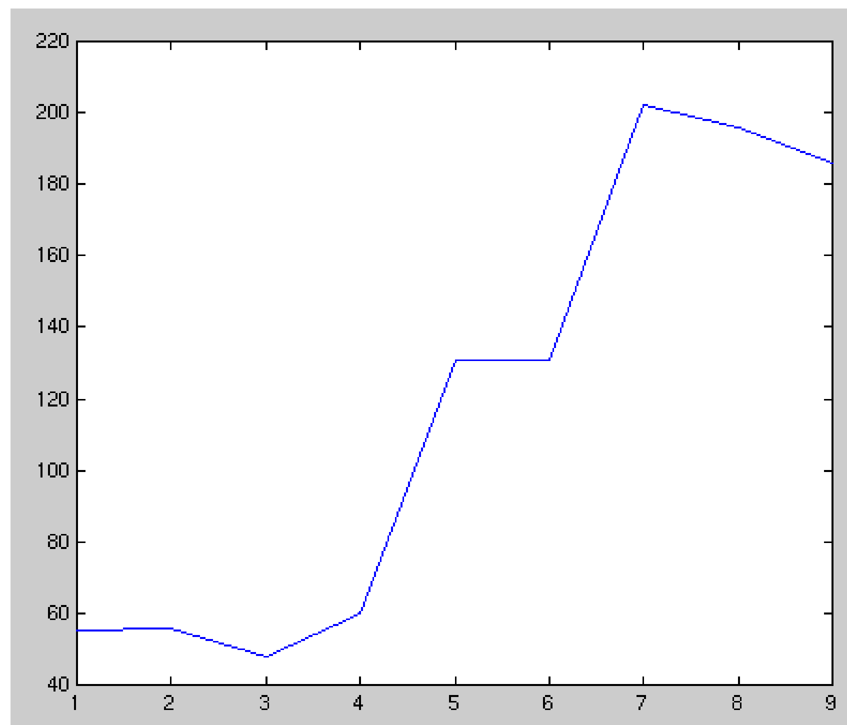
- Edge Detection
  - Gradient operators
  - Canny edge detectors
  - Laplacian detectors



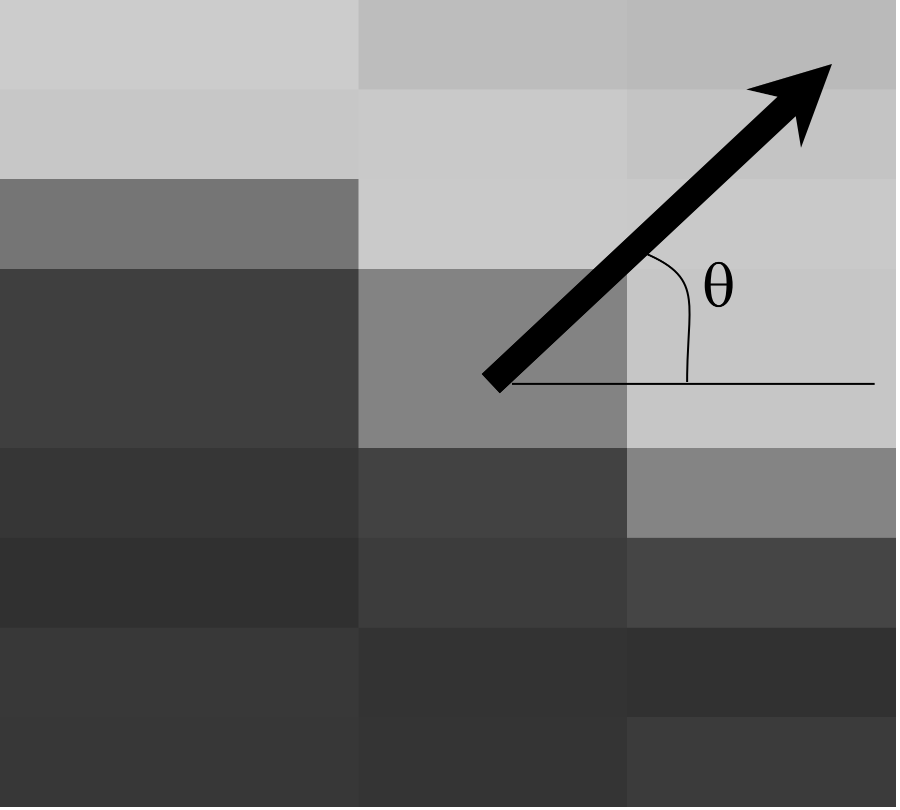


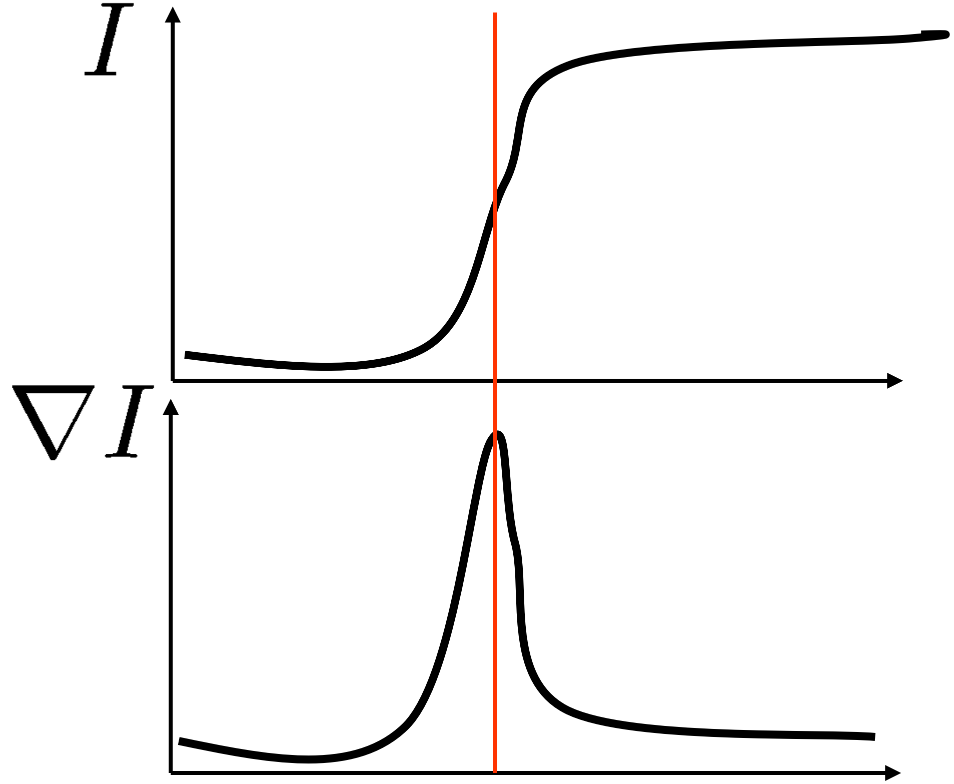
What is an edge?





Edge = discontinuity of intensity in some direction.
Could be detected by looking for places where the derivatives of the image have large values.

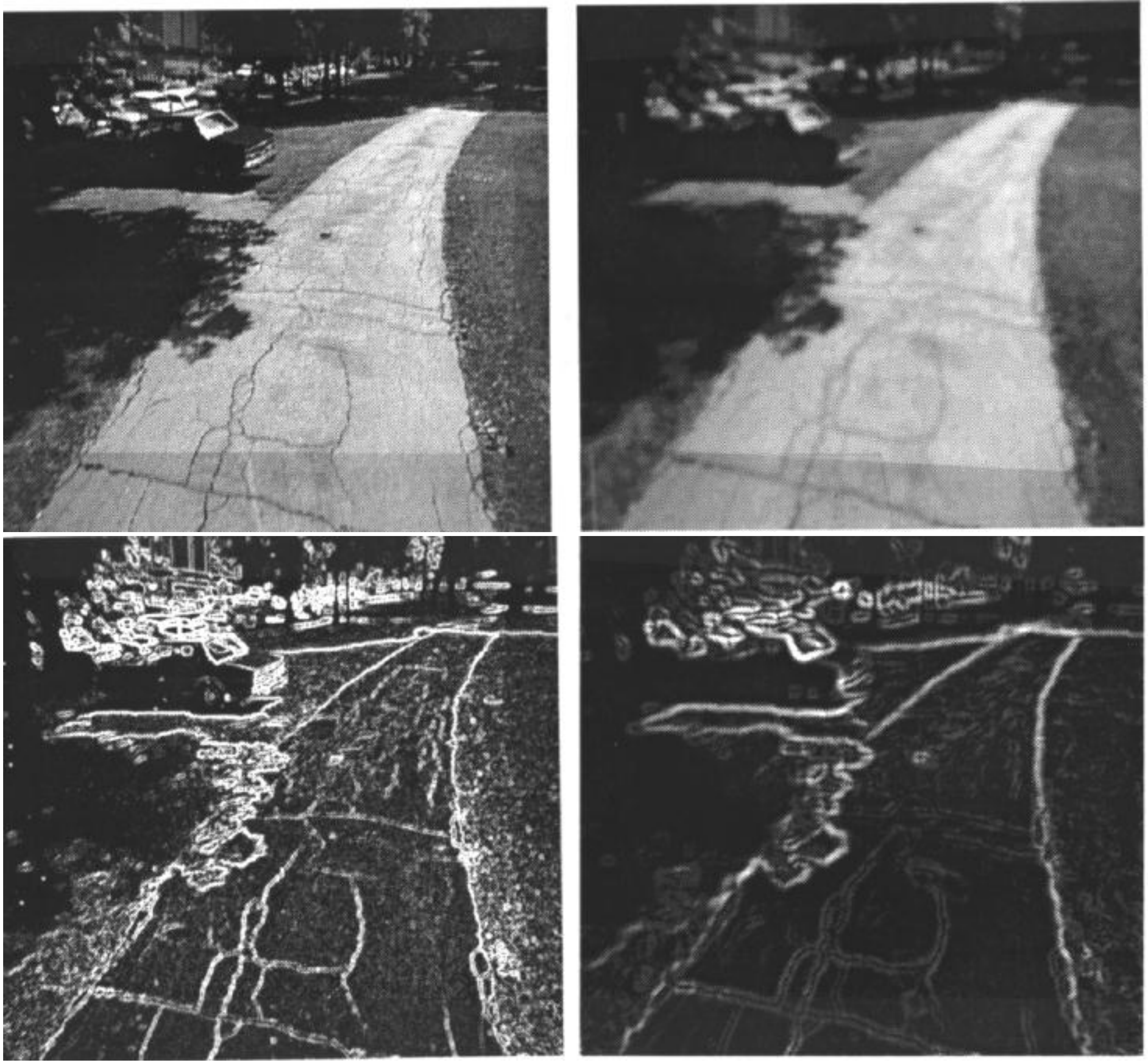




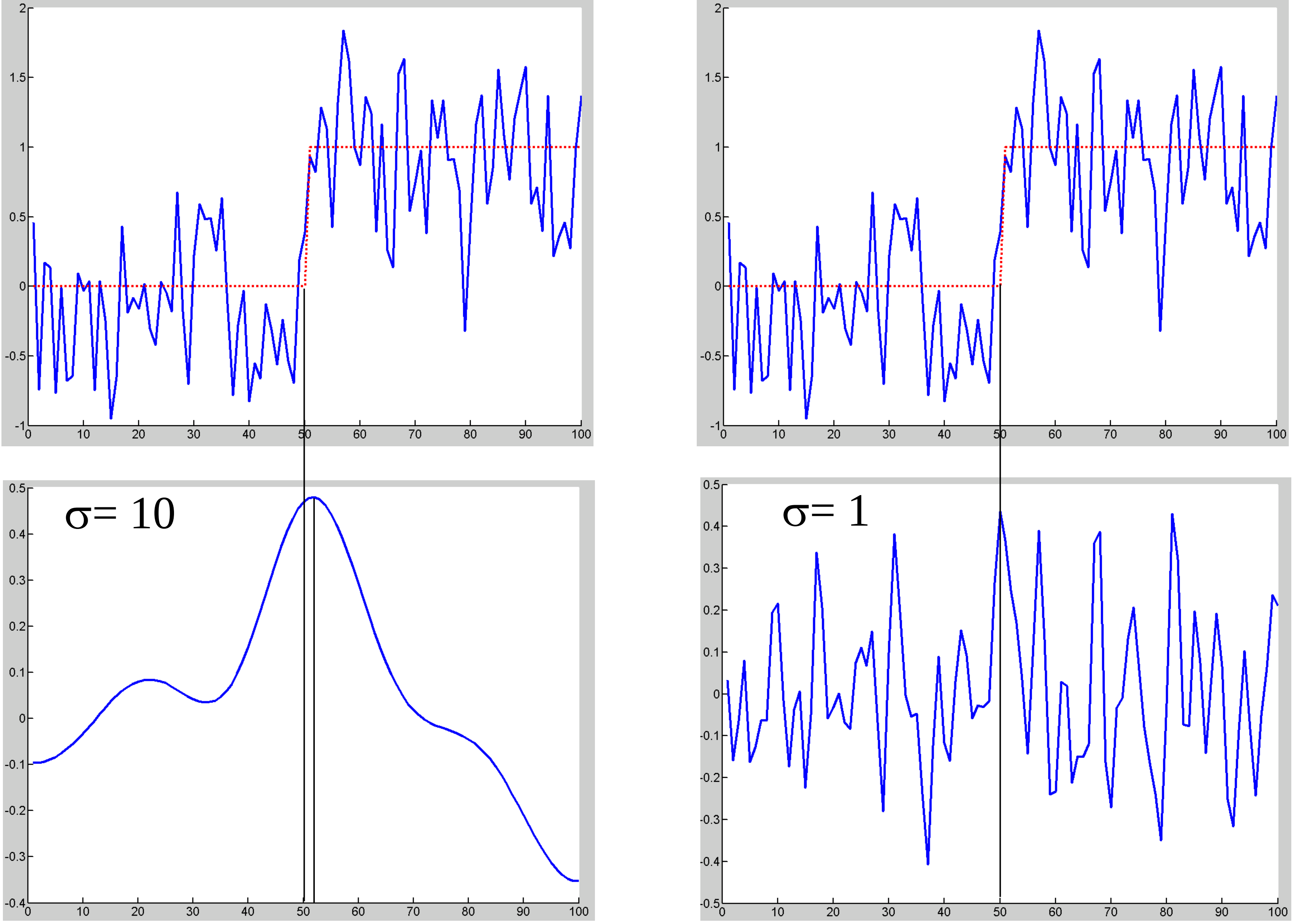
Edge pixels are at local maxima of gradient magnitude
Gradient computed by convolution with Gaussian derivatives
Gradient direction is always perpendicular to edge direction

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

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



Small sigma Large sigma

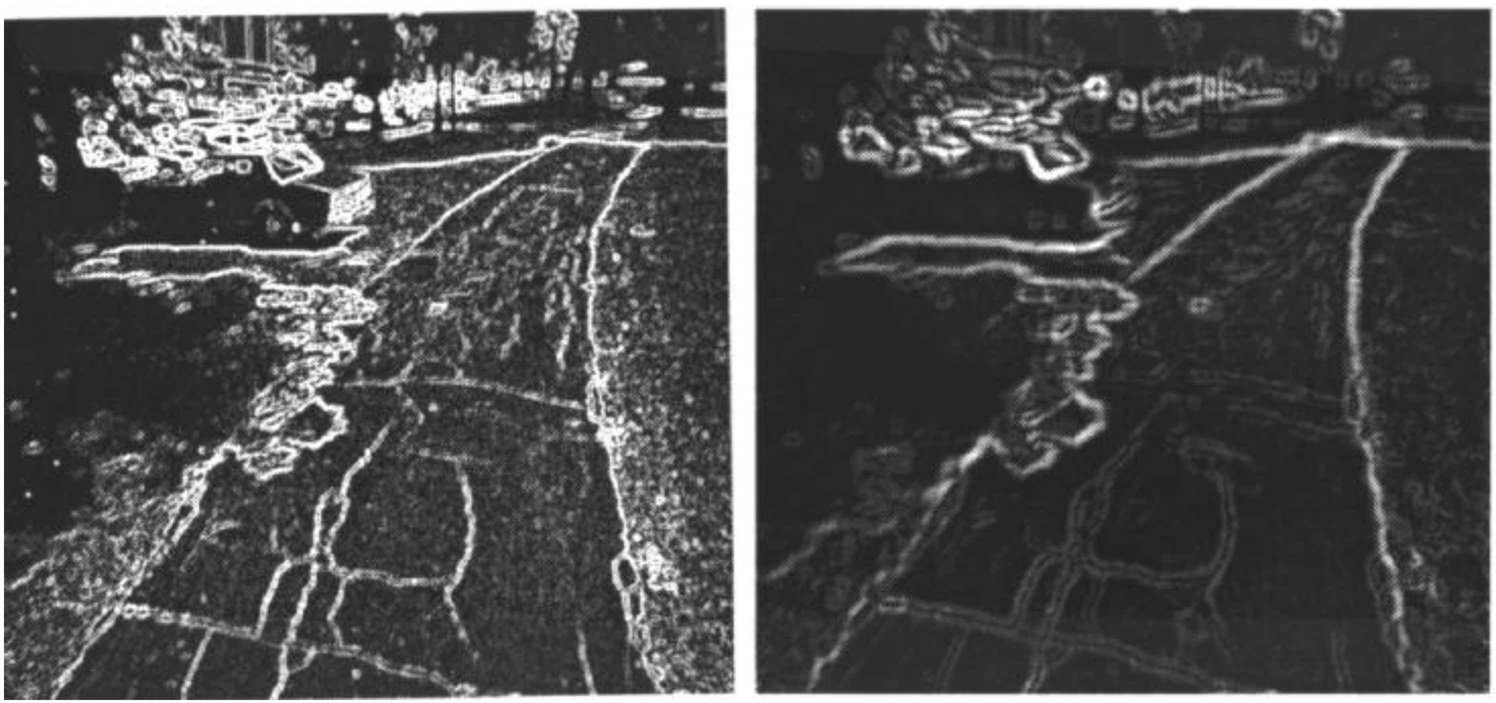


Large σ → Good detection (high SNR)
Poor localization

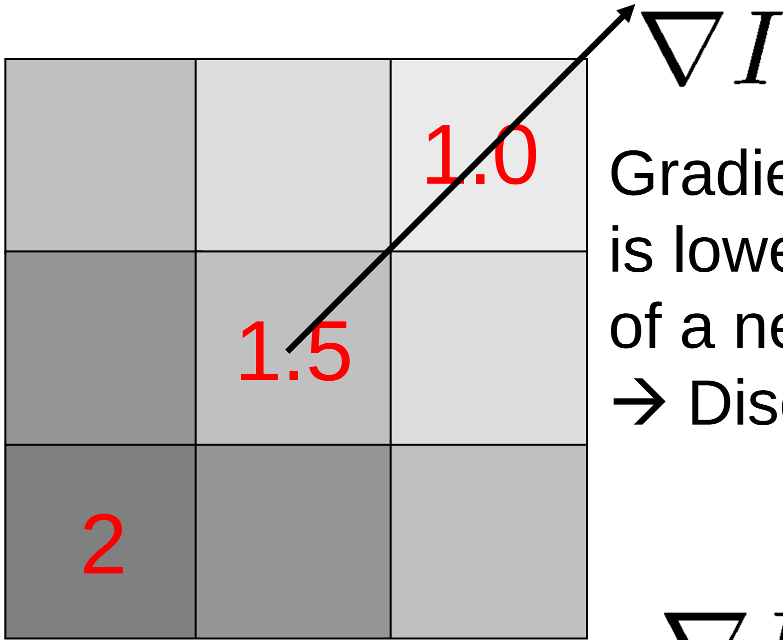
Small σ → Poor detection (low SNR)
Good localization

Next Steps

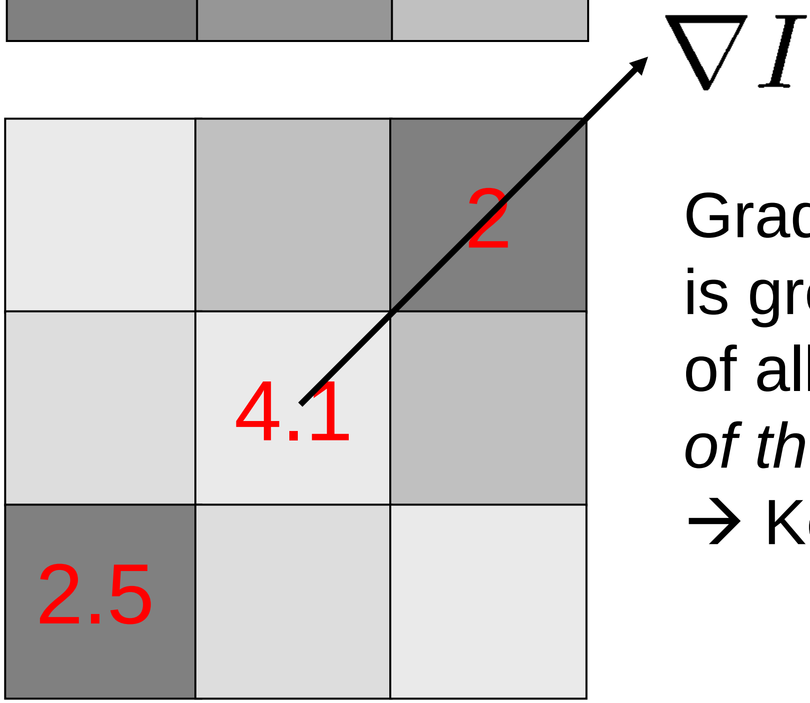
- The gradient magnitude enhances the edges but problems remain:
  – Even if we had a perfect threshold, we would still have poorly localized edges. How to extract optimally localize contours?
- Solution:
  – Non-local maxima suppression



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

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

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 σ: Poor localization, good detection
  - Small σ: Good localization, poor detection
- Gaussian derivatives yield good compromise between localization and detection