Introduction à la vision artificielle V

Jean Ponce
Email: ponce@di.ens.fr
Web: http://www.di.ens.fr/~ponce

Planches après les cours sur :
http://www.di.ens.fr/~ponce/introvis/lect5.pptx
http://www.di.ens.fr/~ponce/introvis/lect5.pdf

Souvenez vous: Le premier exo est du aujourd'hui..
Du le 20:
http://www.di.ens.fr/willow/teaching/introvis14/assignment2/

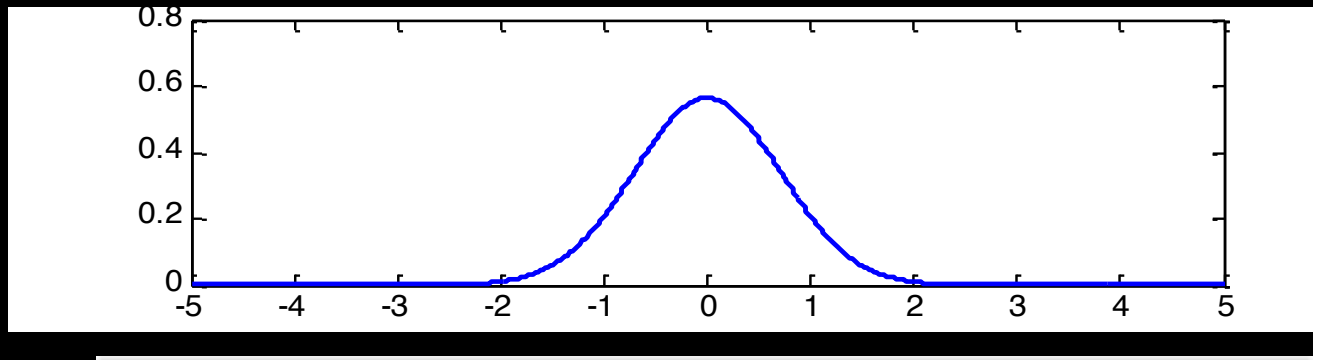
Gaussian Filtering and Denoising

- Gaussian filters and noise
- Separability
- Oriented filters
- Sparse coding and noise

Gaussian filters

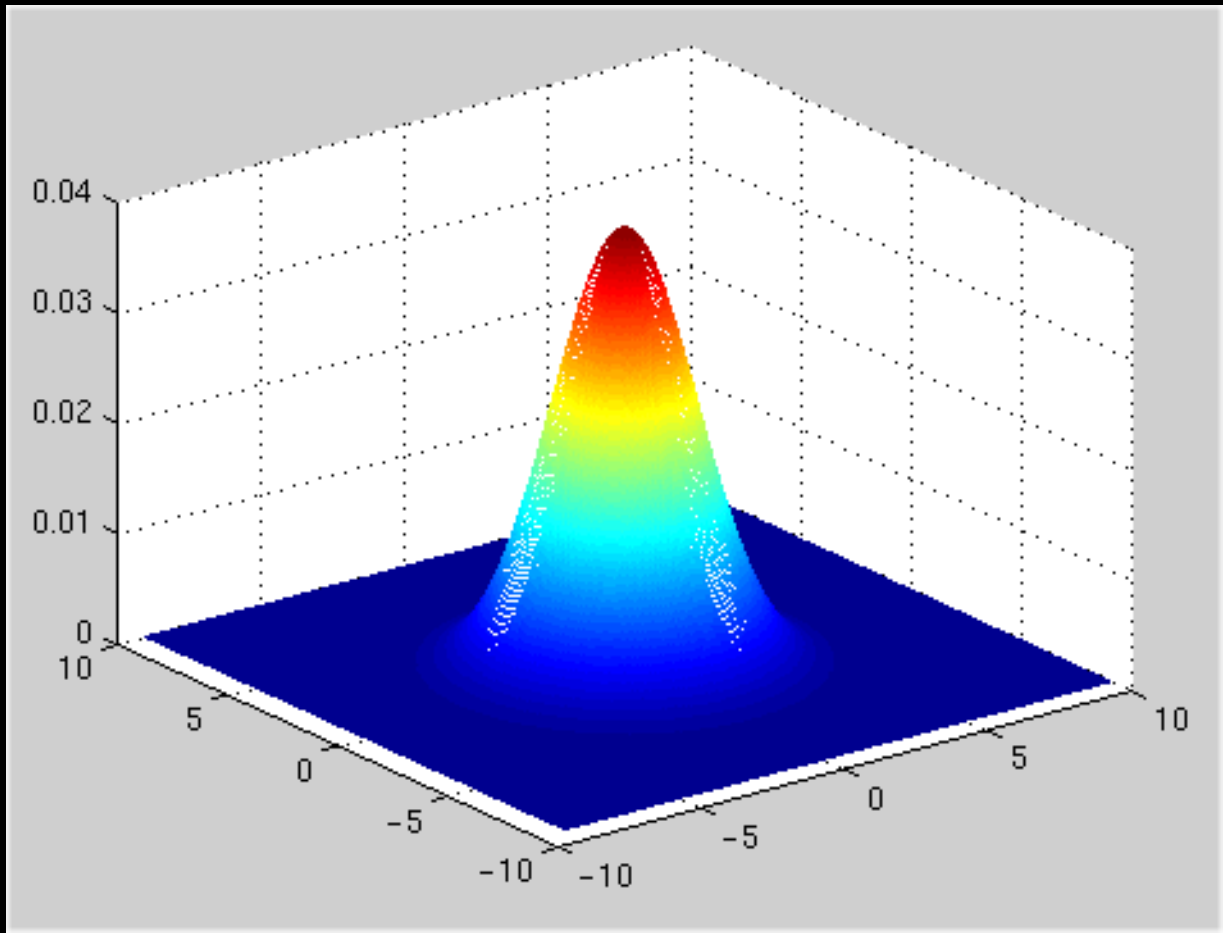
1-D:

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



2-D:

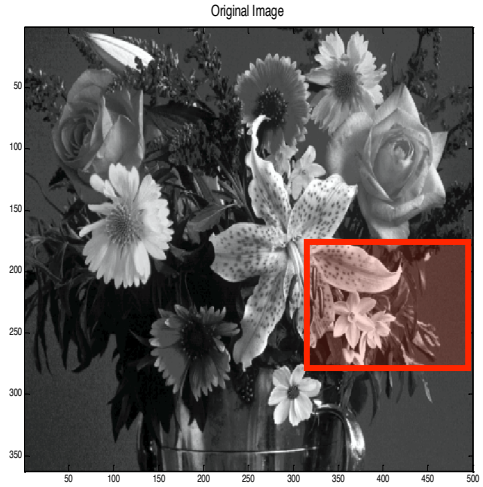
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

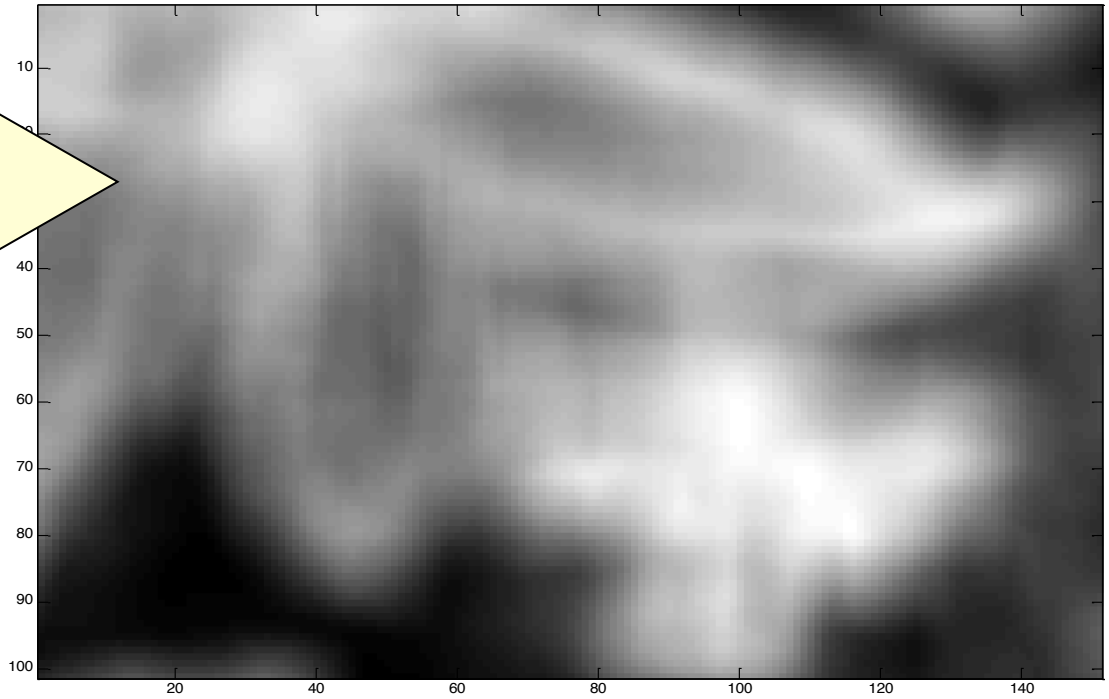


Slight abuse of notation:
We ignore the normalization constant such that

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

Simple Averaging





Gaussian Smoothing

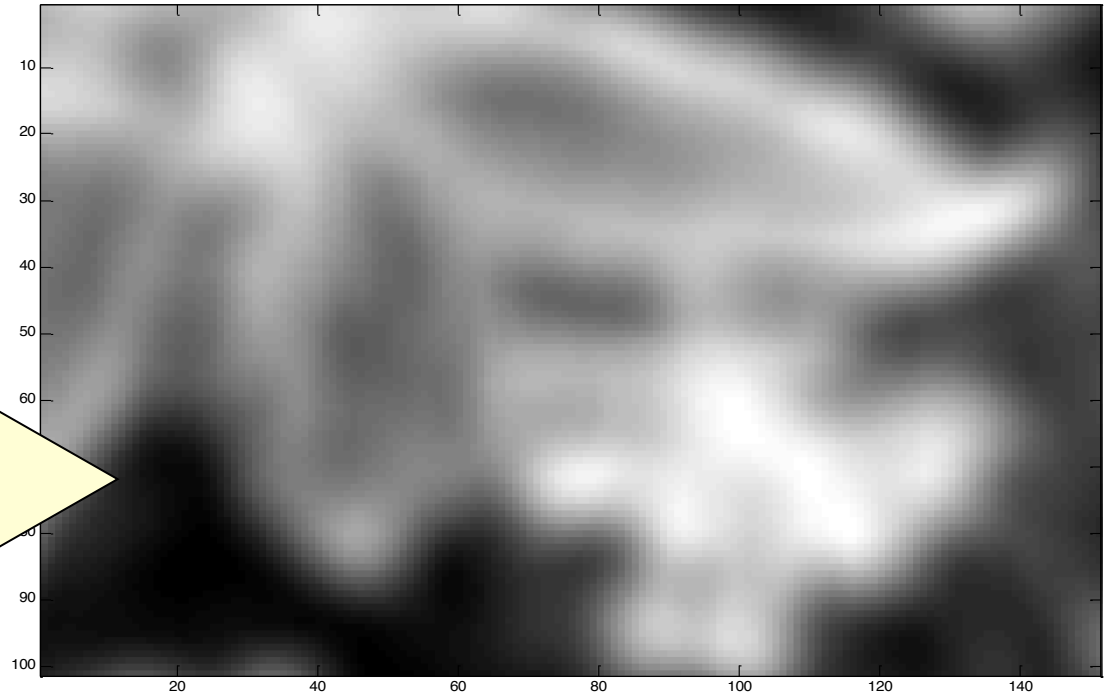
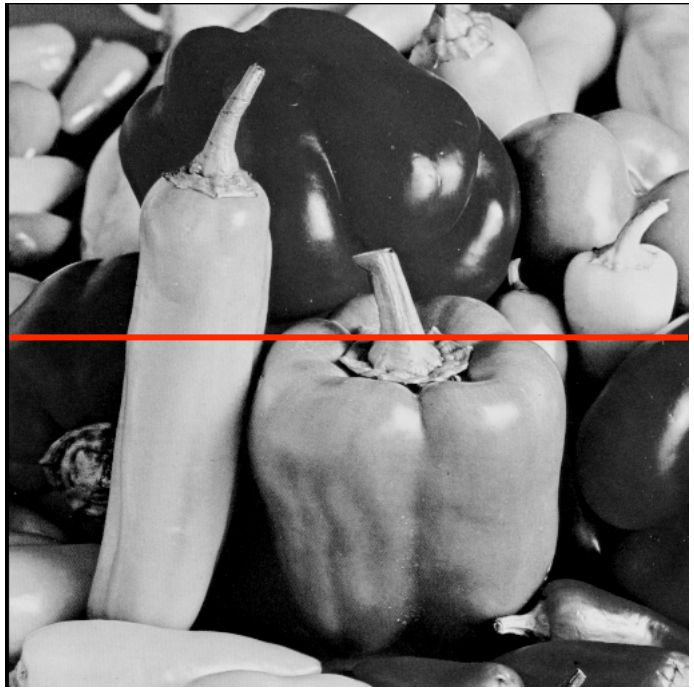
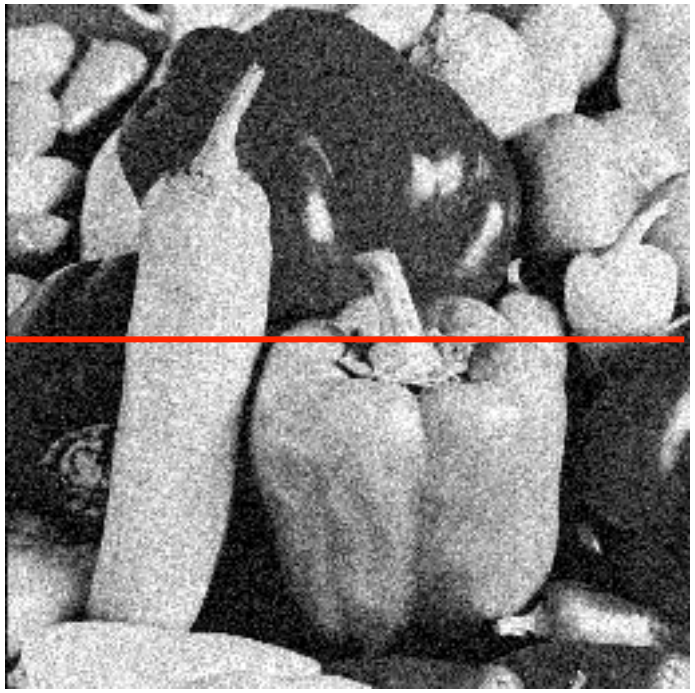
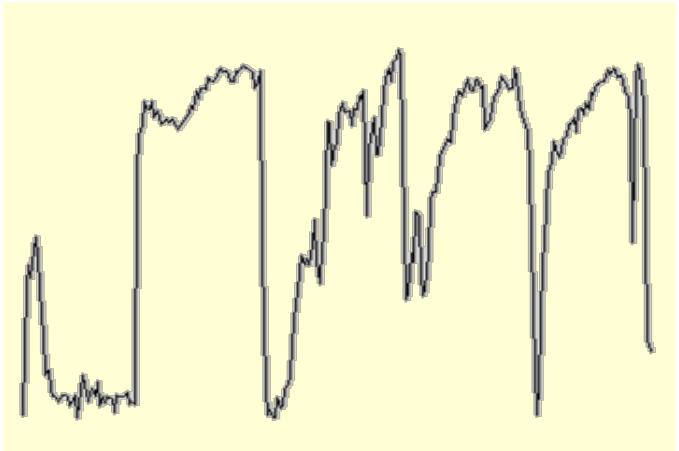
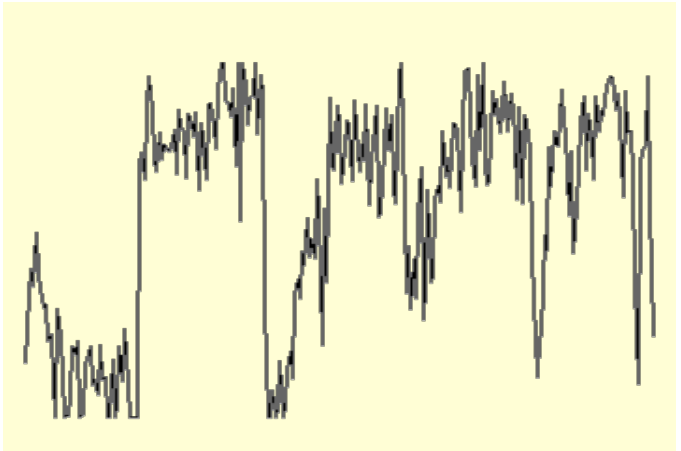


Image Noise







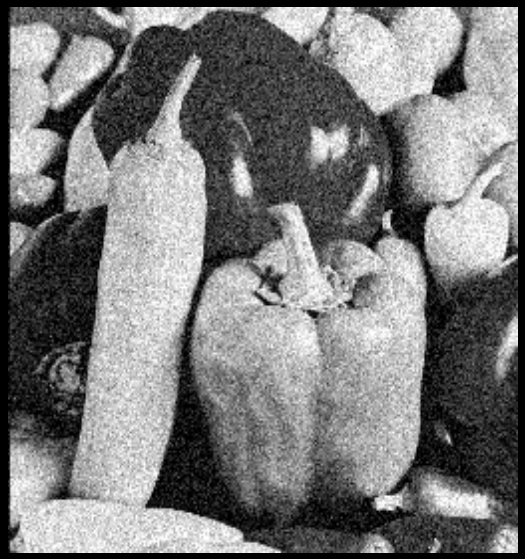


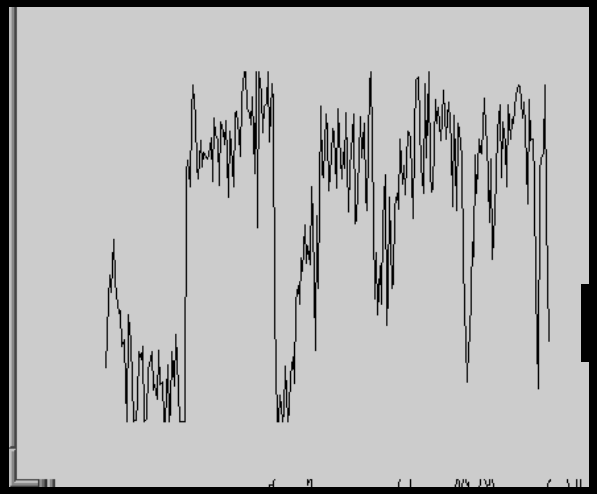
$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

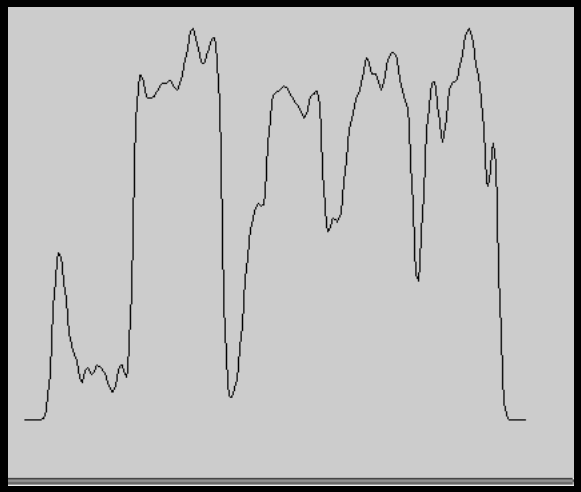
Gaussian i.i.d. ("white") noise:

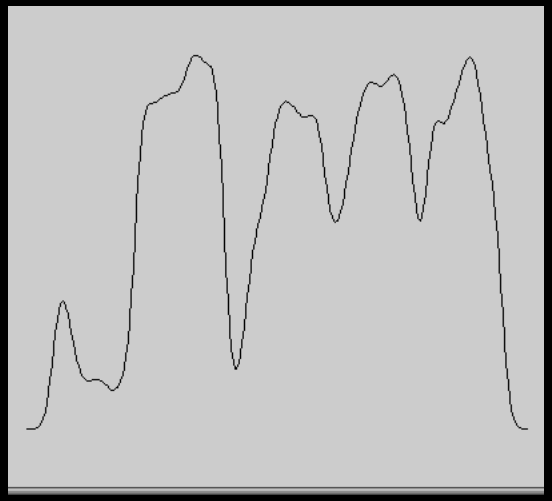
$$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$$

Gaussian Smoothing to Remove Noise









No smoothing

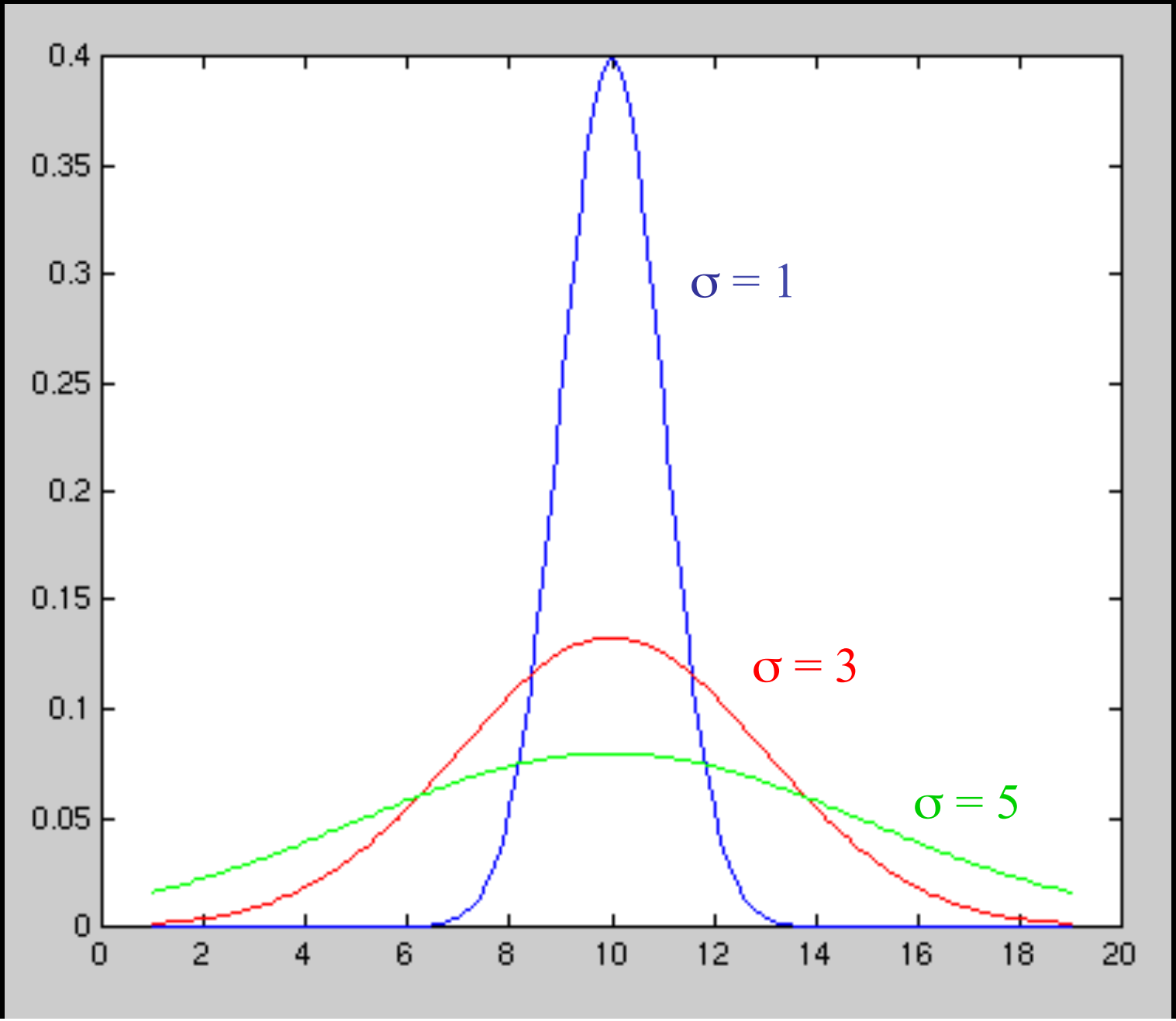
$\sigma = 2$

$\sigma = 4$

Bottom line: The standard deviation of white noise is divided by k*sigma

Increasing σ

Shape of Gaussian filter as function of s



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:

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

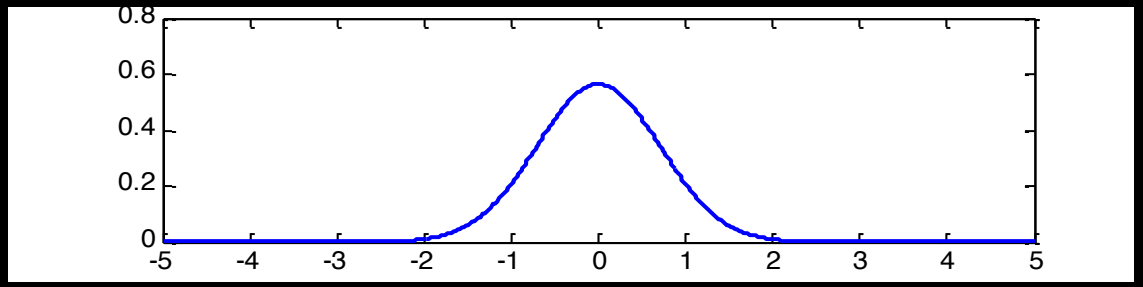
- Separable filter:

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

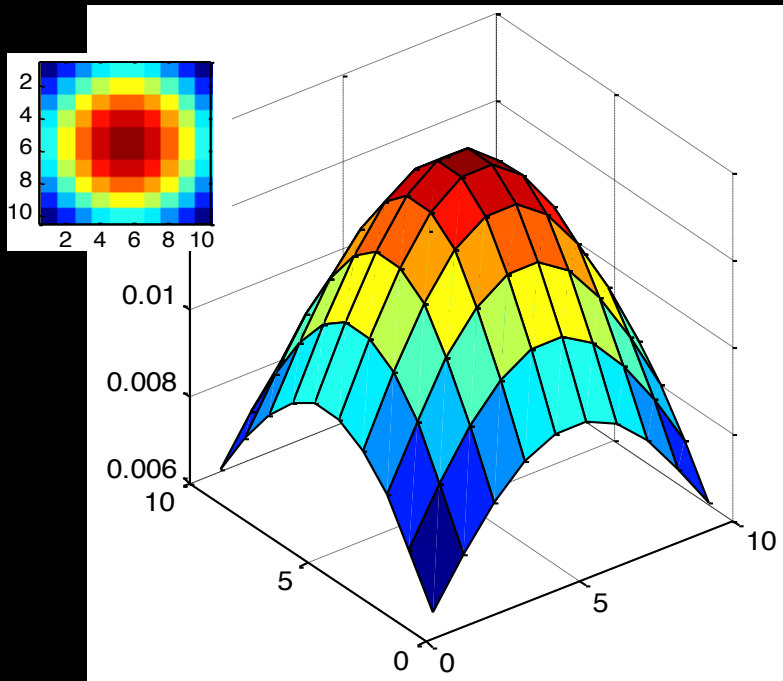
- Critical implication: Filtering with a *N*x*N* Gaussian kernel can be implemented as two convolutions of size *N* → reduction quadratic to linear → *must* be implemented that way

Note about Finite Kernel Support

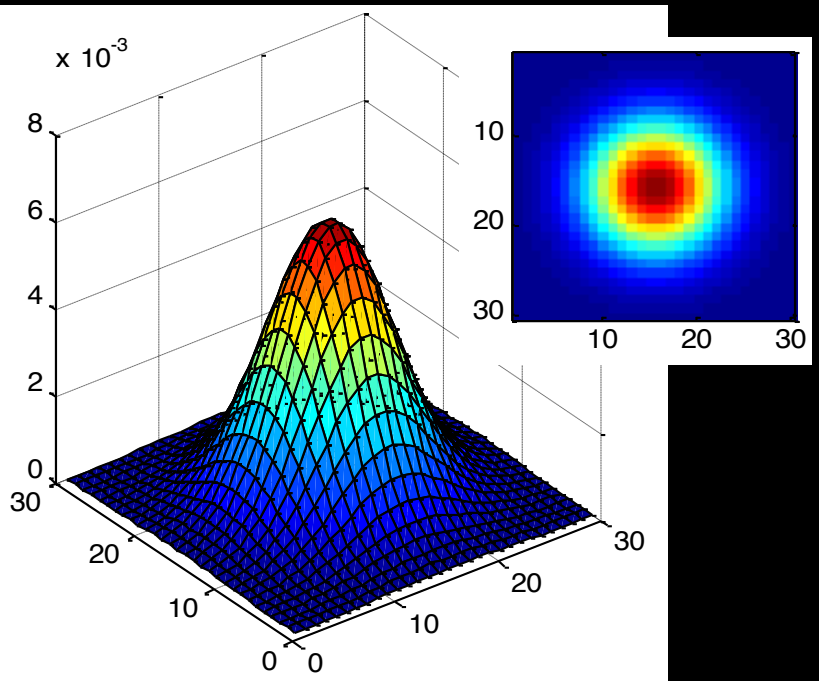
- Gaussian function has infinite support



- In actual filtering, we have a finite kernel size



σ = 5 with 10x10 kernel

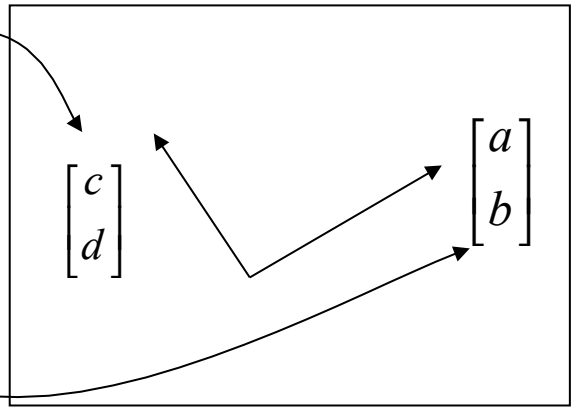


σ = 5 with 30x30 kernel

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]

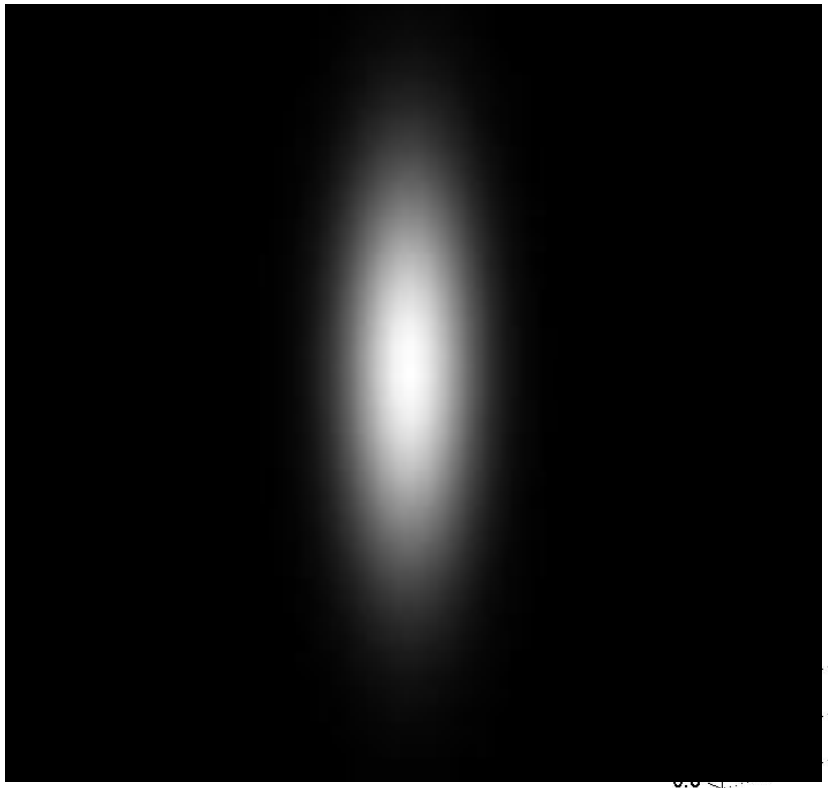
$$G = e^{-\frac{(ax+by)^2}{2\sigma_1^2} - \frac{(cx+dy)^2}{2\sigma_2^2}}$$



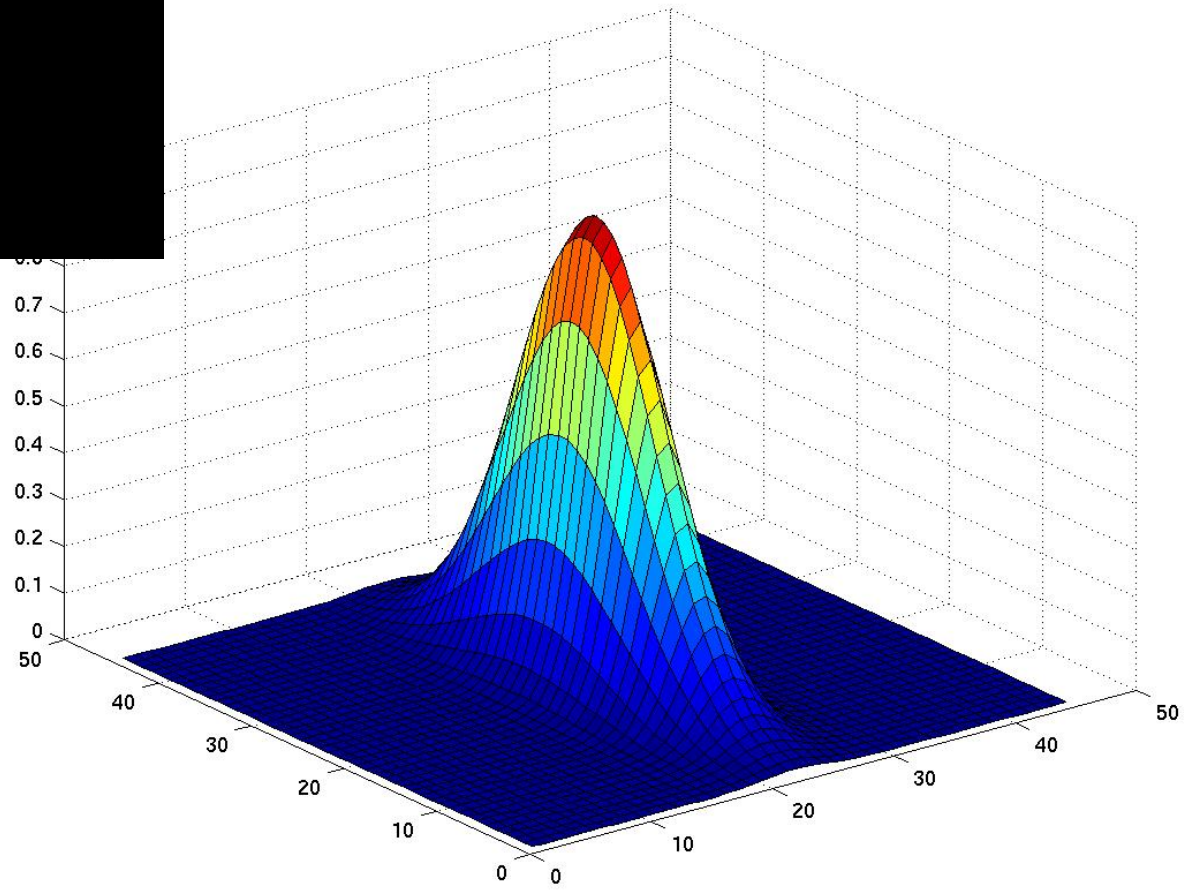
- We can write this in a more compact form by using the standard multivariate Gaussian notation:

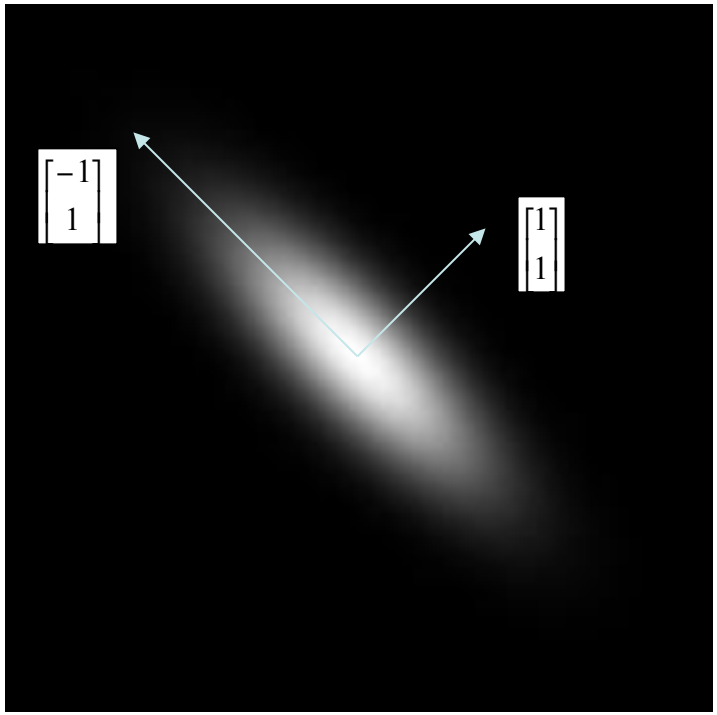
$$G_\Sigma = e^{-\frac{X^T \Sigma^{-1} X}{2}} \qquad X = \begin{bmatrix} x \\ y \end{bmatrix}$$

- 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.



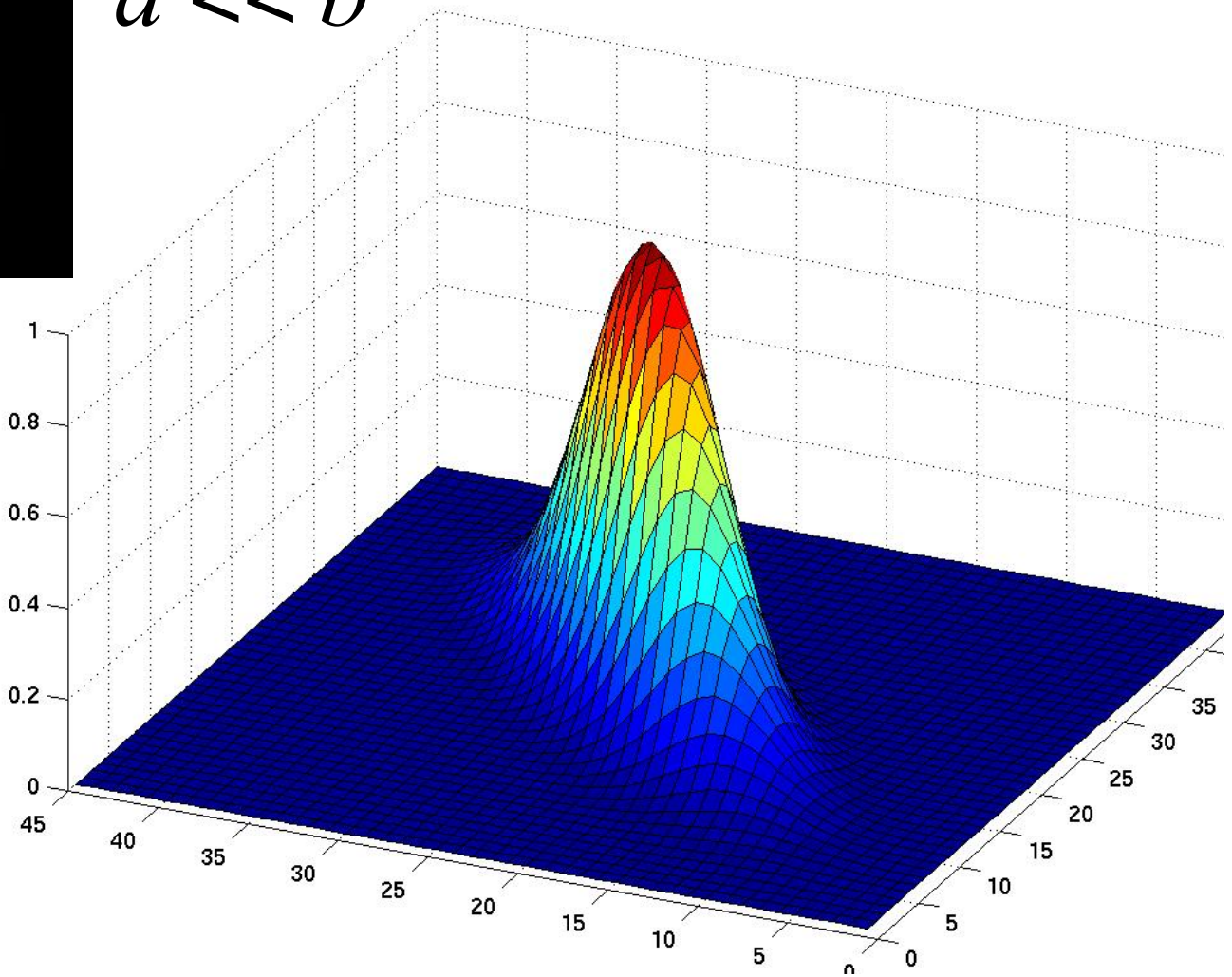
$$\Sigma = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad a << b$$

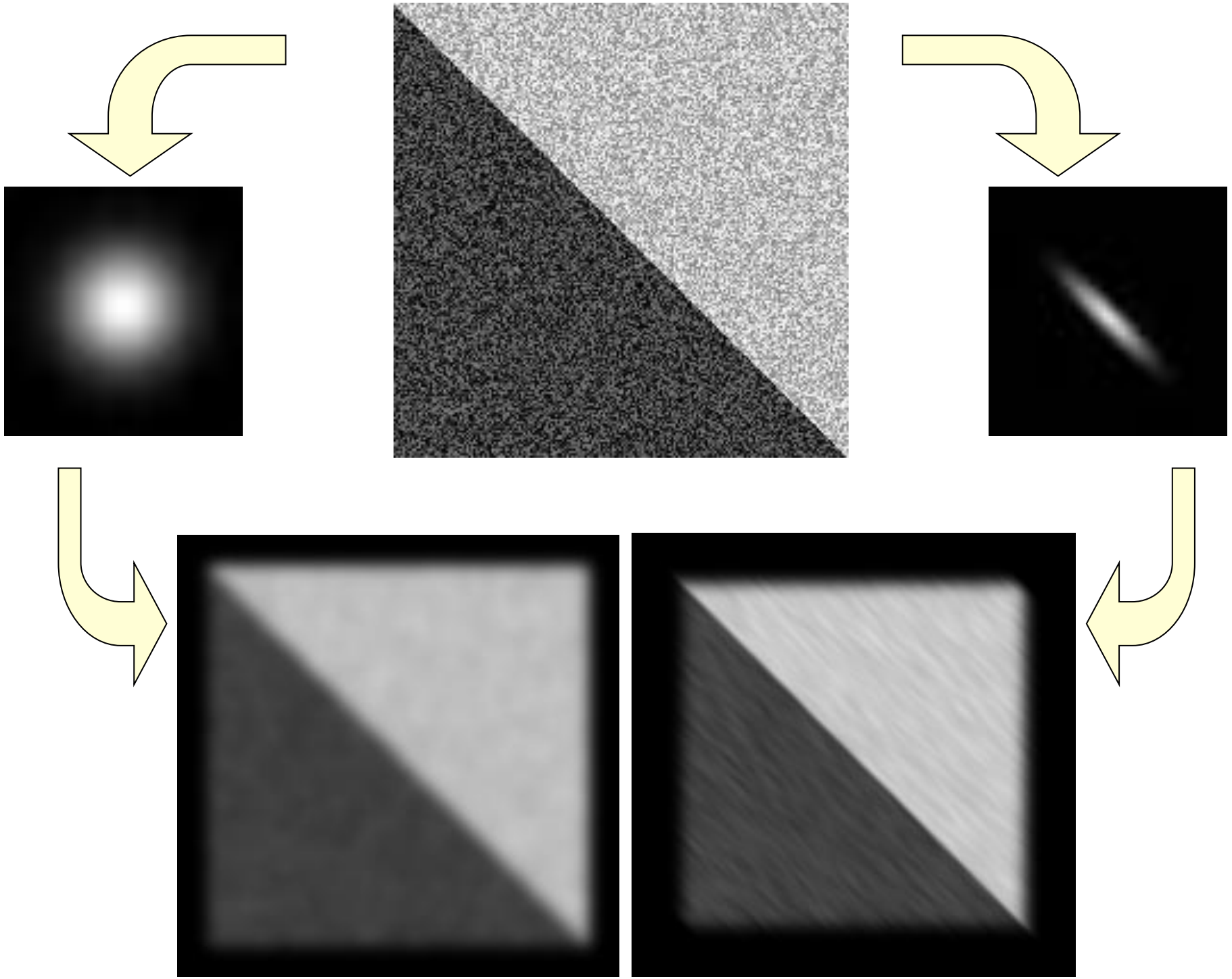




$$\Sigma = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

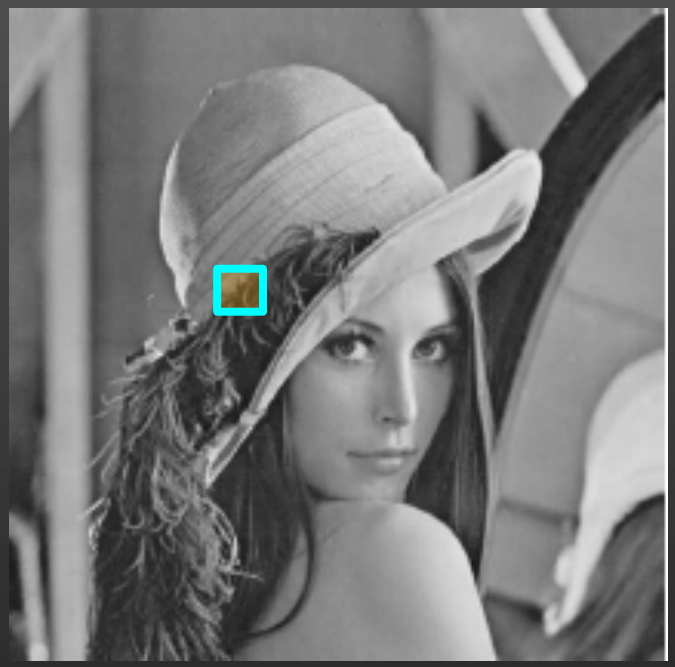
$$a << b$$





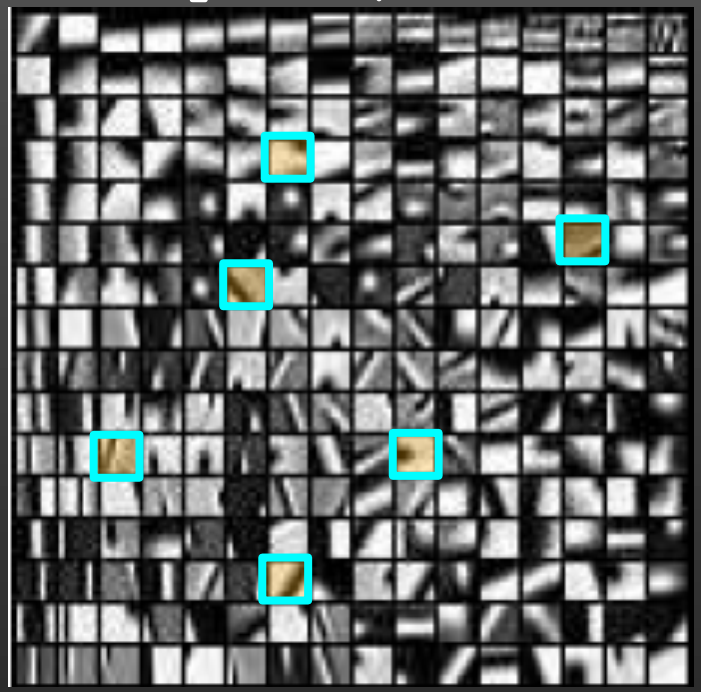
Sparse linear models

Signal: $x \in R^m$



Dictionary:

$D=[d_1,\ldots,d_p] \in R^{m \times p}$

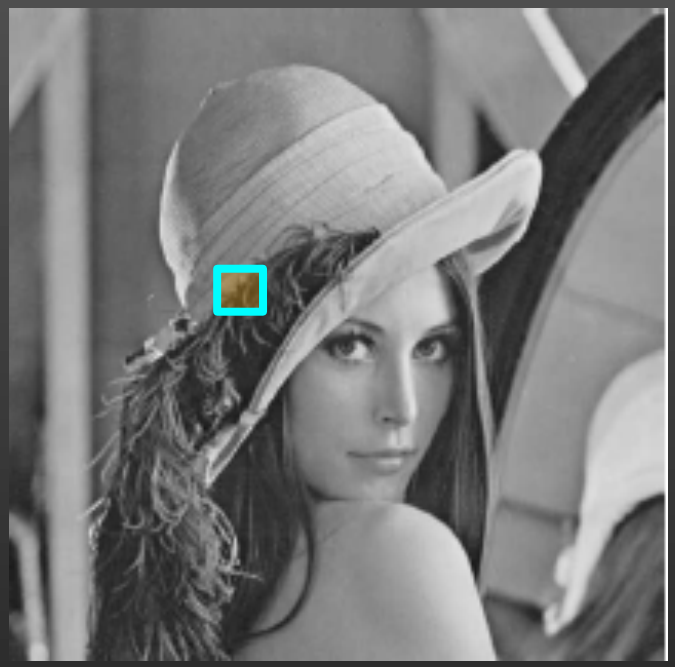


$$x \approx \alpha_1 d_1 + \alpha_2 d_2 + \ldots + \alpha_p d_p = D\alpha, \text{ with } |\alpha|_0 \ll p$$

(Olshausen and Field, 1997; Chen et al., 1999; Mallat, 1999; Elad and Aharon, 2006)
(Kavukcuoglu et al., 2009; Wright et al., 2009; Yang et al., 09; Boureau et al., 2010)

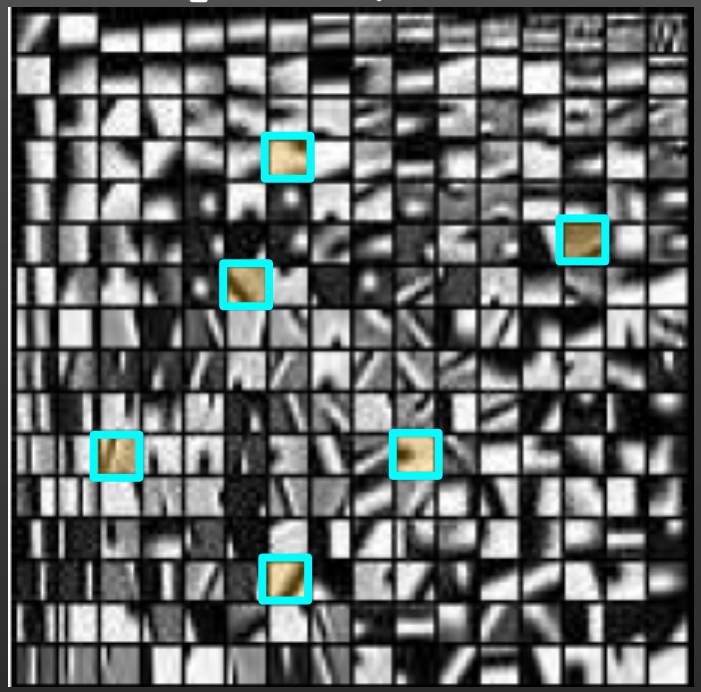
Sparse linear models

Signal: $x \in \mathbb{R}^m$



Dictionary:

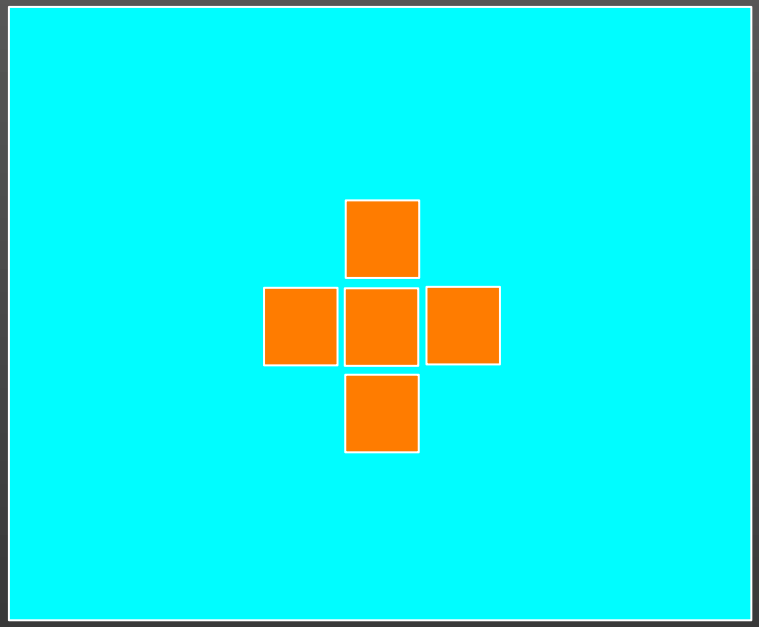
$D=[d_1,\ldots,d_p] \in \mathbb{R}^{m \times p}$

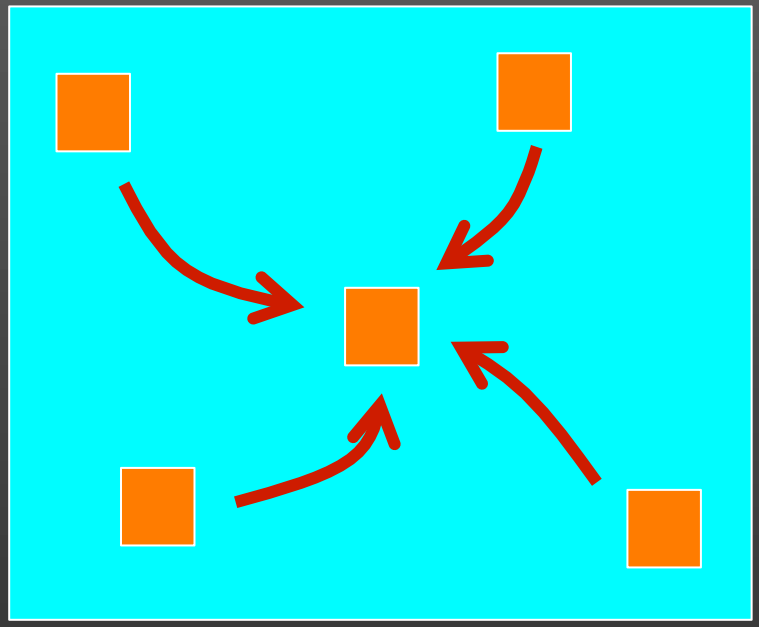


$$x \approx \alpha_1 d_1 + \alpha_2 d_2 + \ldots + \alpha_p d_p = D\alpha, \text{ with } |\alpha|_1 \ll \rho$$

(Olshausen and Field, 1997; Chen et al., 1999; Mallat, 1999; Elad and Aharon, 2006)
(Kavukcuoglu et al., 2009; Wright et al., 2009; Yang et al., 09; Boureau et al., 2010)

State of the art in image denoising





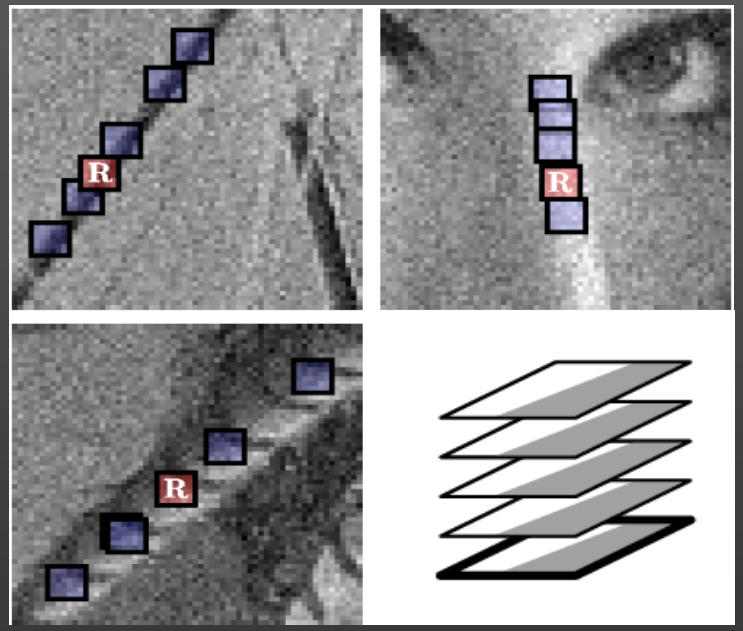
Non-local means filtering (Buades et al.'05)

Dictionary learning for denoising (Elad & Aharon'06; Mairal, Elad & Sapiro'08)

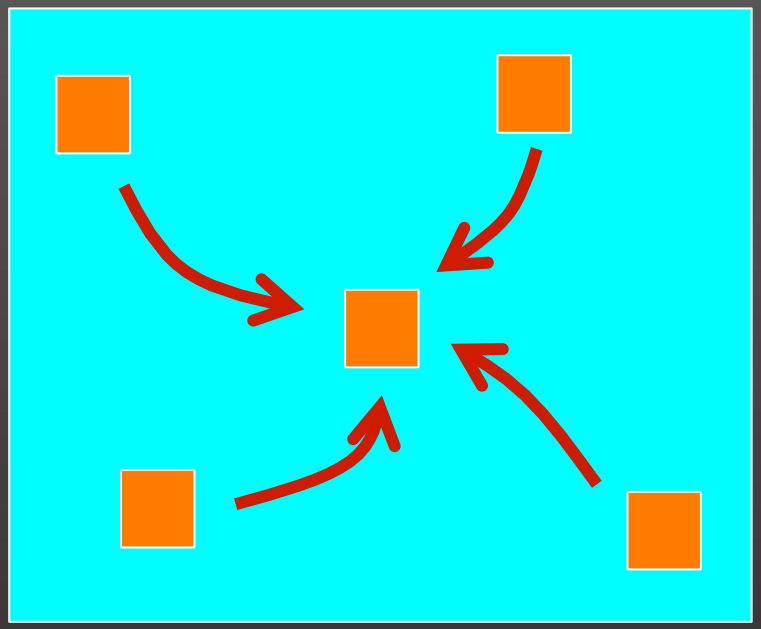
$$\min_{D \in C, \alpha_1, \ldots, \alpha_n} \sum_{1 \le i \le n} [\, 1/2 \, | \, x_i - D\alpha_i \, |_2^2 + \lambda \, |\alpha_i|_1 \,]$$

$$x = 1/n \sum_{1 \le i \le n} R_i D \alpha_i$$

State of the art in image denoising



BM3D (Dabov et al.'07)



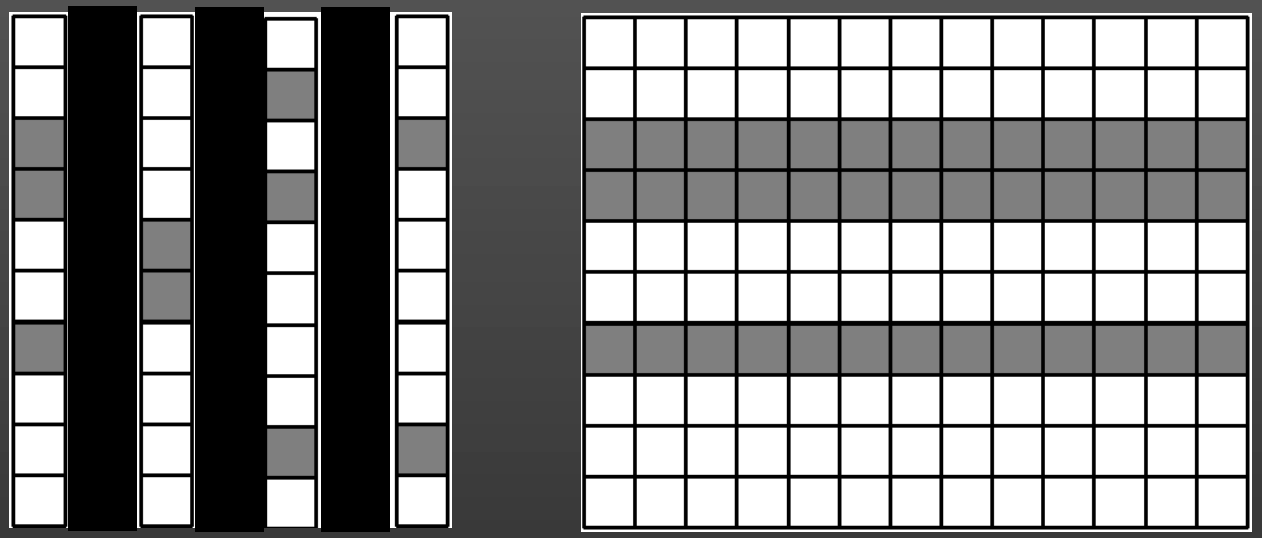
Non-local means filtering (Buades et al.'05)

Dictionary learning for denoising (Elad & Aharon'06; Mairal, Elad & Sapiro'08)

$$\min_{D \in C, \alpha_1, \ldots, \alpha_n} \sum_{1 \le i \le n} [\, 1/2 \, | \, x_i - D\alpha_i \, |_2^2 + \lambda \, |\alpha_i|_1 \,]$$

$$x = 1/n \sum_{1 \le i \le n} R_i D\alpha_i$$

Non-local sparse models for image restoration (Mairal, Bach, Ponce, Sapiro, Zisserman, ICCV'09)



Sparsity vs Joint sparsity

$$\min_{\substack{D\in C \\ A_1,\ldots,A_n}} \sum_i \Big[\sum_{j\in S_i} 1/2 \,|\, x_j - D\alpha_{ij} \,|_F^2\Big] + \lambda \,|A_i|_{p,q}$$

$$|A|_{p,q} = \sum_{1\le i\le k} |\alpha^i|_q^p \quad (p,q) = (1,2) \text{ or } (0,\infty)$$

σ	[23]	[25]	[12]	[8]	SC	LSC	LSSC
5	37.05	37.03	37.42	37.62	37.46	37.66	**37.67**
10	33.34	33.11	33.62	34.00	33.76	33.98	**34.06**
15	31.31	30.99	31.58	32.05	31.72	31.99	**32.12**
20	29.91	29.62	30.18	30.73	30.29	30.60	**30.78**
25	28.84	28.36	29.10	29.72	29.18	29.52	**29.74**
50	25.66	24.36	25.61	26.38	25.83	26.18	**26.57**
100	22.80	21.36	22.10	23.25	22.46	22.62	**23.39**

PSNR comparison between our method (LSSC) and Portilla et al.'03 [23]; Roth & Black'05 [25]; Elad& Aharon'06 [12]; and Dabov et al.'07 [8].

Real noise (Canon Powershot G9, 1600 ISO)

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

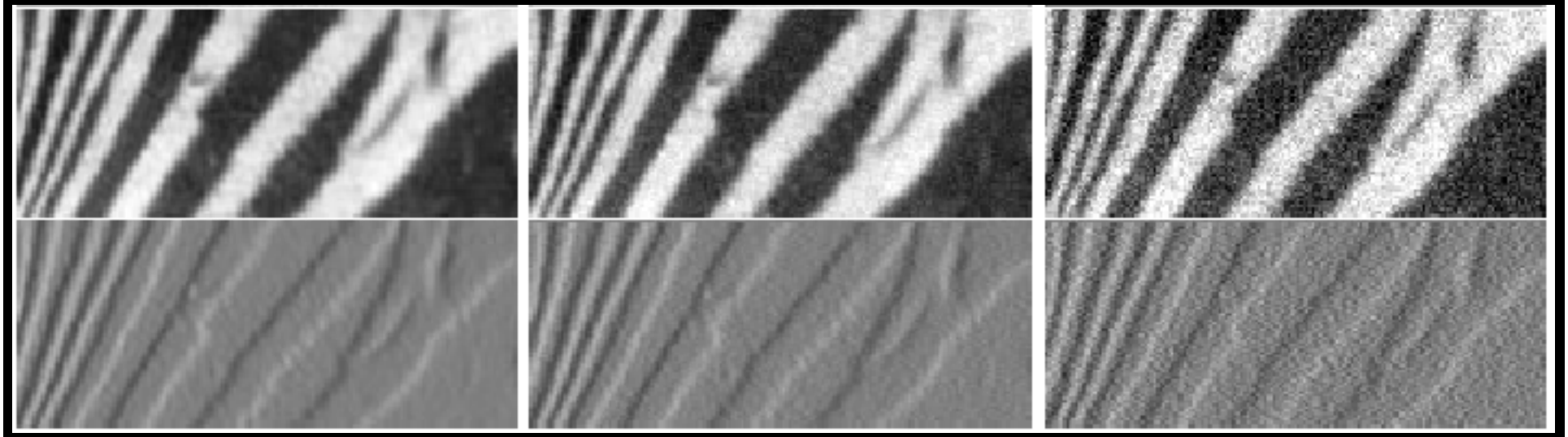
$$\partial_x = \boxed{1}\boxed{0}\boxed{-1}$$

- Problem: Increases noise

$$\underbrace{I(i+1, j) - I(i-1, j)}_{\text{Difference between Actual image values}} = \underbrace{\hat{I}(i+1, j) - \hat{I}(i-1, j)}_{\text{True difference (derivative)}} + \underbrace{n_+ + n_-}_{\text{Sum of the noises}}$$

Finite differences

Finite differences responding to noise



Increasing zero-mean Gaussian noise

Smooth Derivatives

- Solution: First smooth the image by a Gaussian G_σ and then take derivatives:

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

- Applying the differentiation property of the convolution:

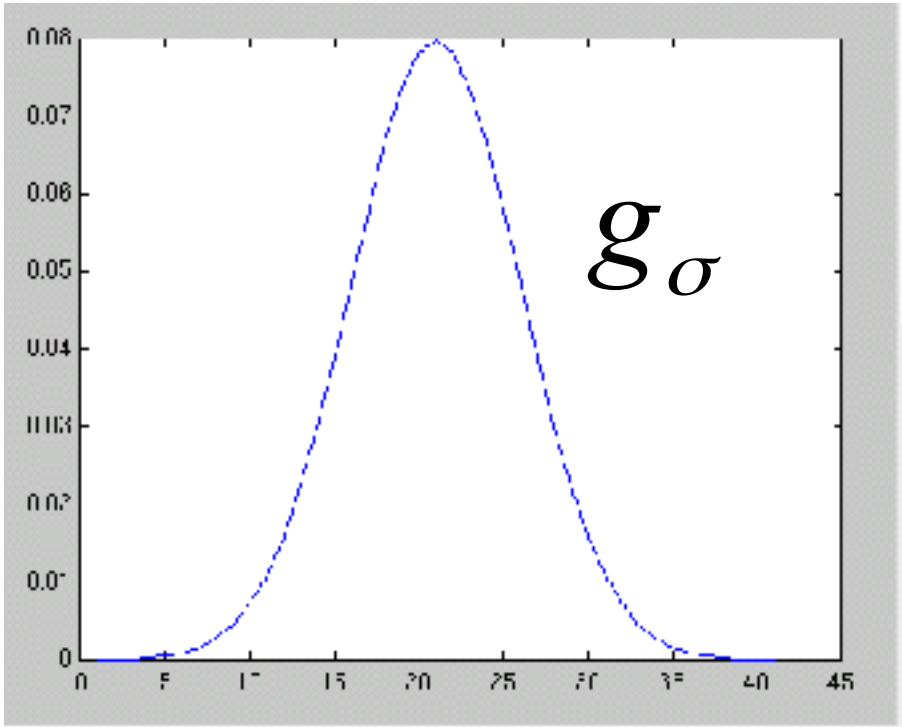
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

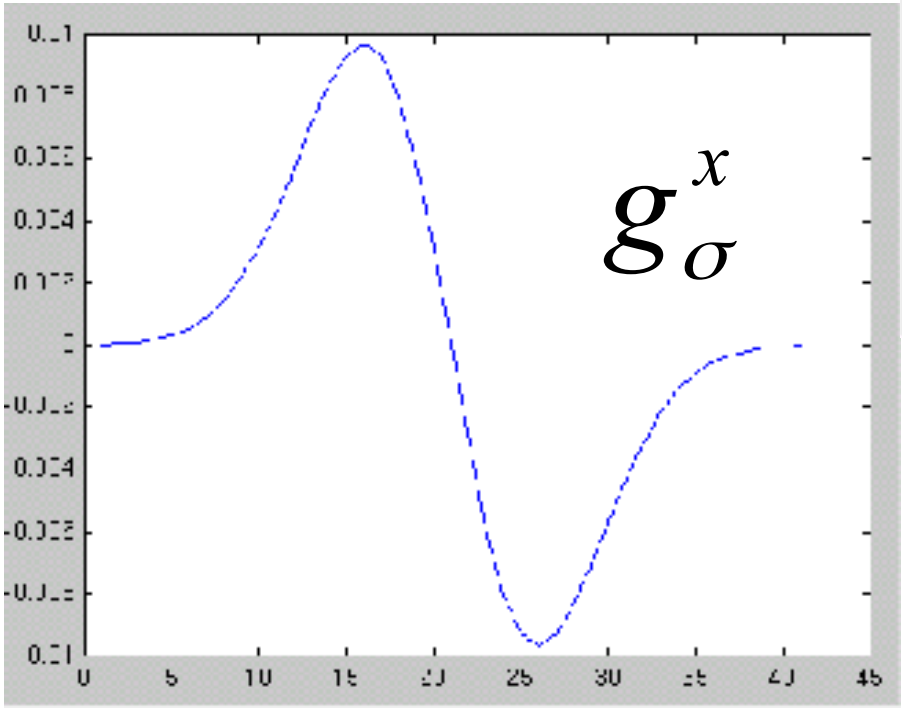
- Therefore, taking the derivative in x of the image can be done by convolution with the derivative of a Gaussian:

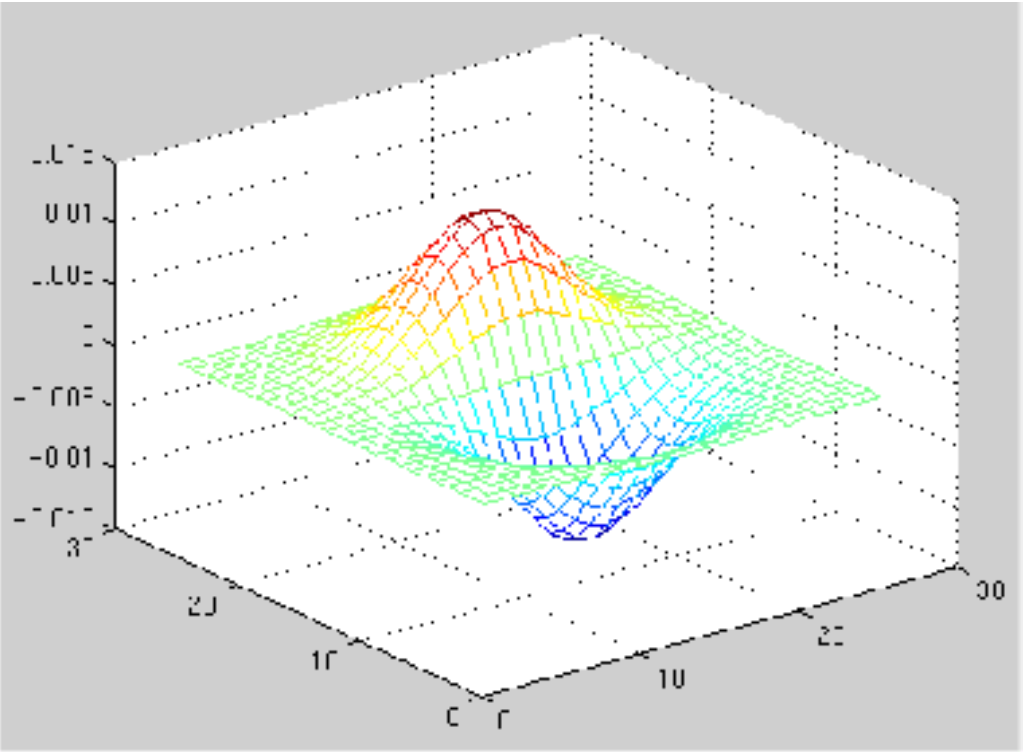
$$G_\sigma^x = \frac{\partial G_\sigma}{\partial x} = xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Crucial property: The Gaussian derivative is also separable:

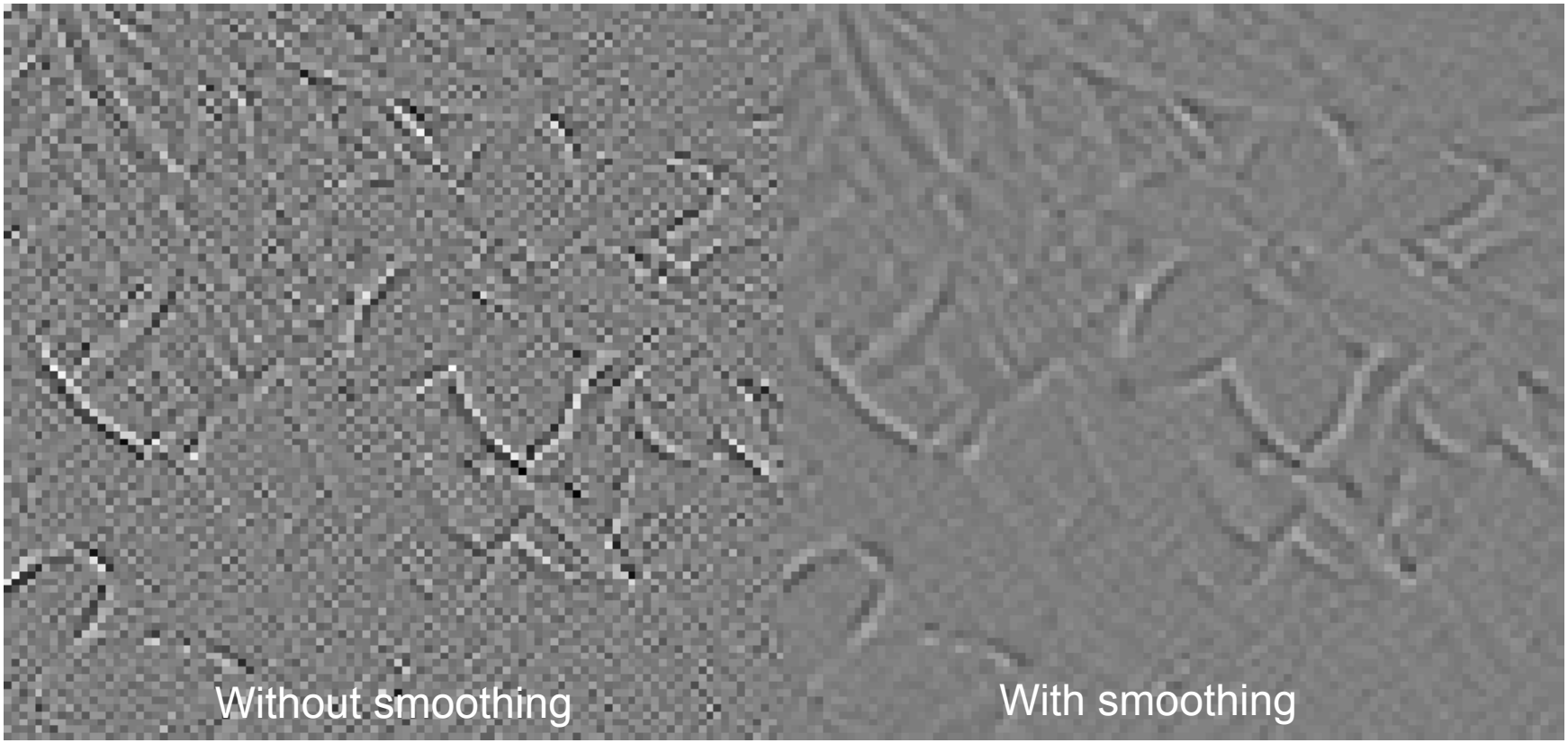
$$G_\sigma^x * f = g_\sigma^x * g_{\sigma\uparrow} * f$$


g_σ


g_σ^x



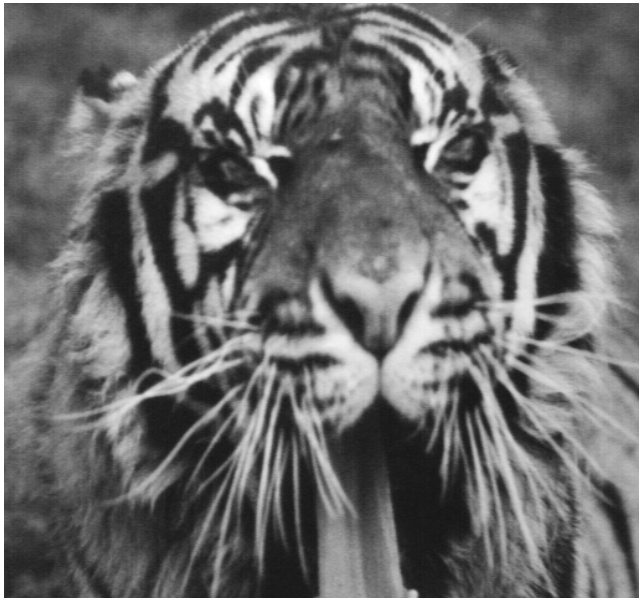
Derivative + Smoothing



Better but still blurs away edge information

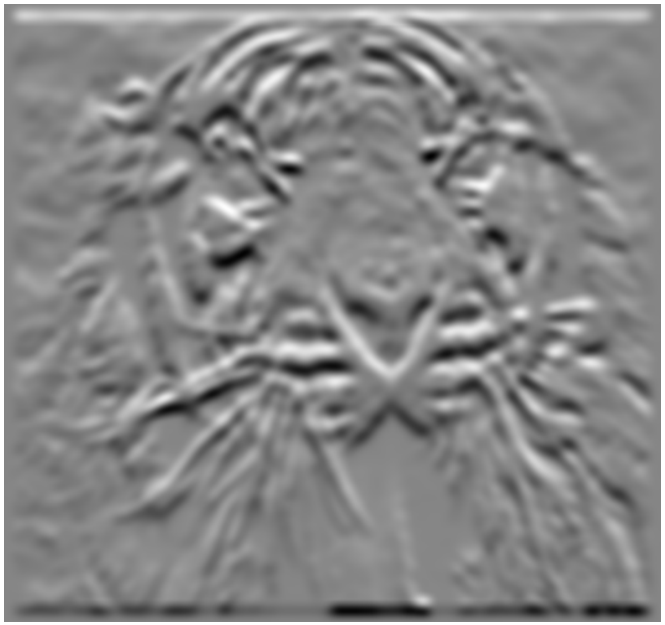
Applying the first derivative of Gaussian

I



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

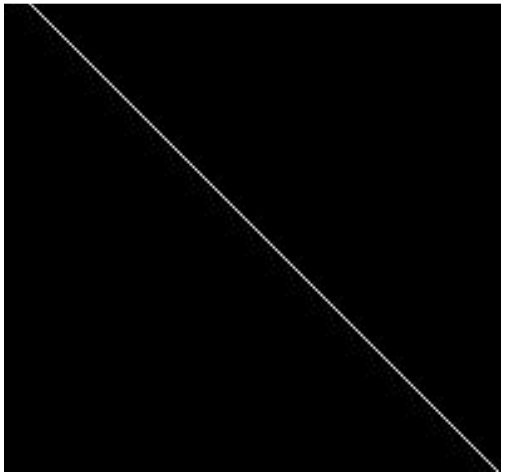
$$\frac{\partial I}{\partial x}$$



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

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

Image with Edge



Edge Location

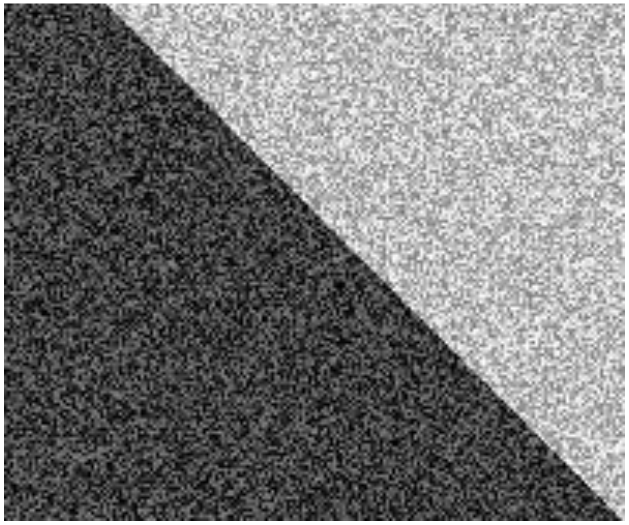
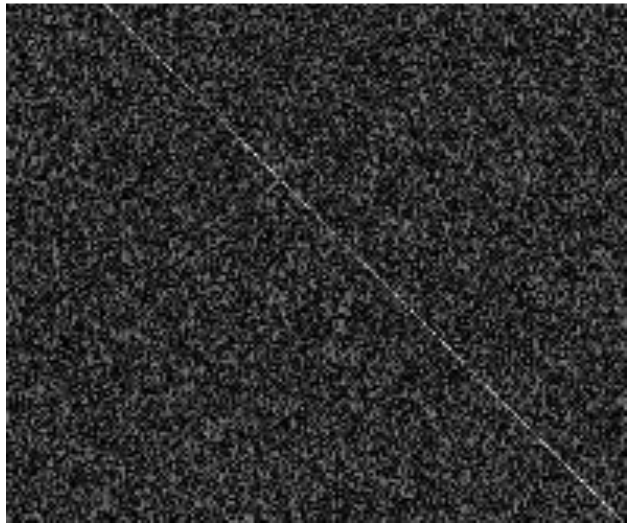
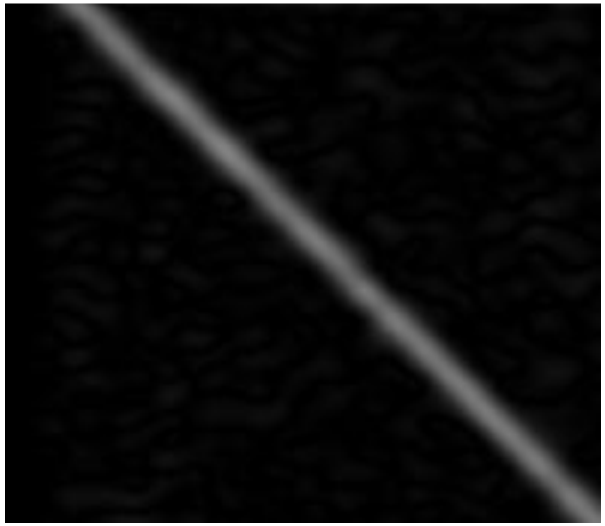


Image + Noise



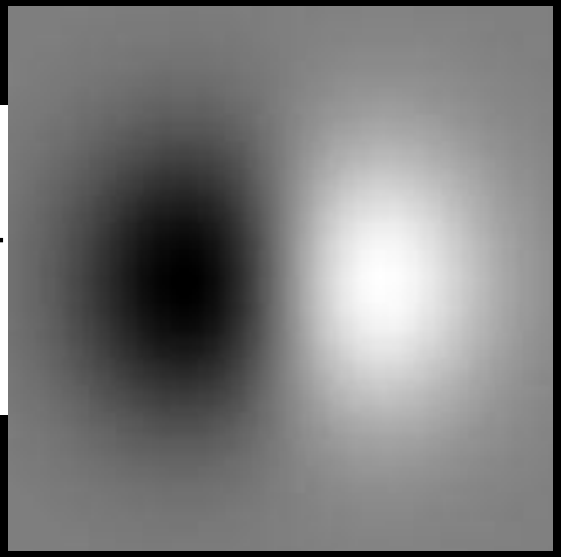
Derivatives detect edge *and* noise



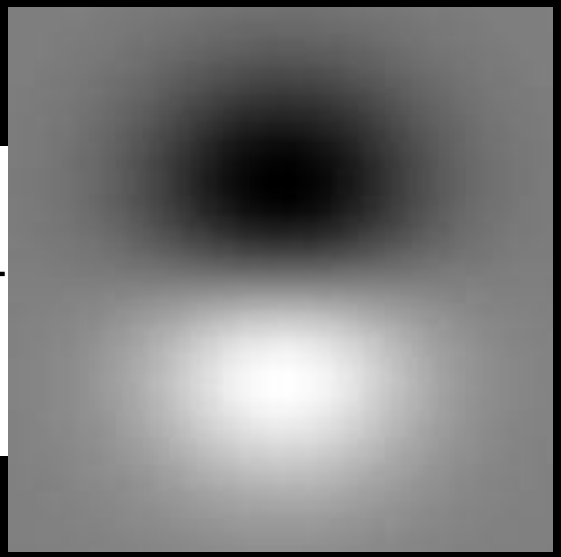
Smoothed derivative removes noise, but blurs edge

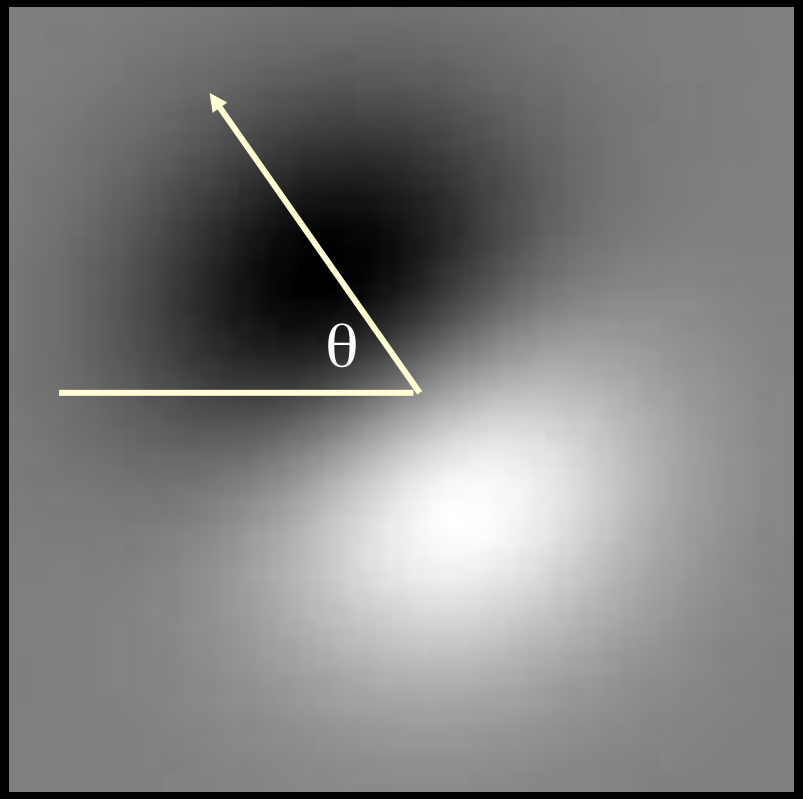
Directional Derivatives

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



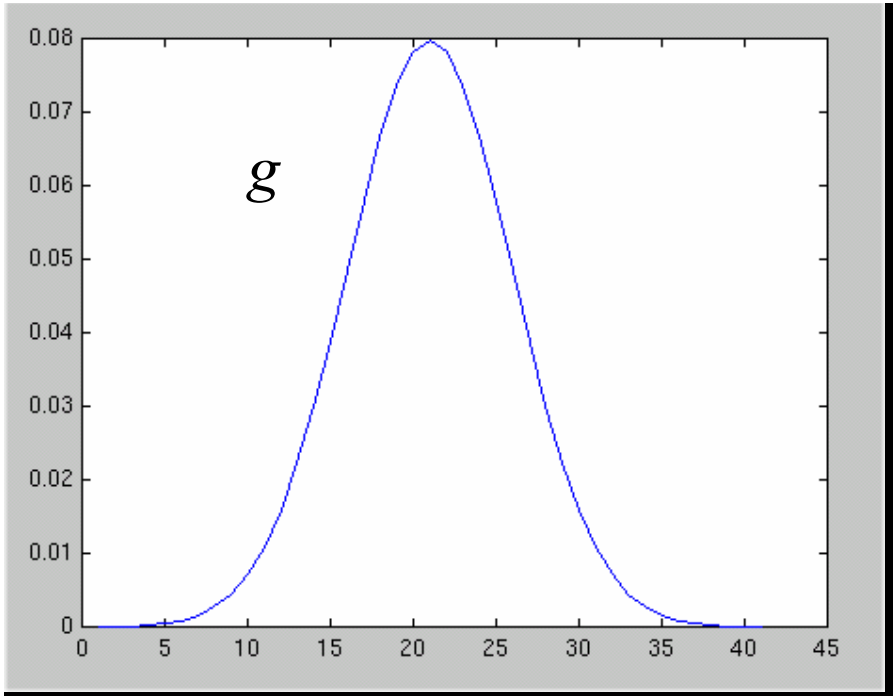
$$\frac{\partial G_\sigma}{\partial y}$$

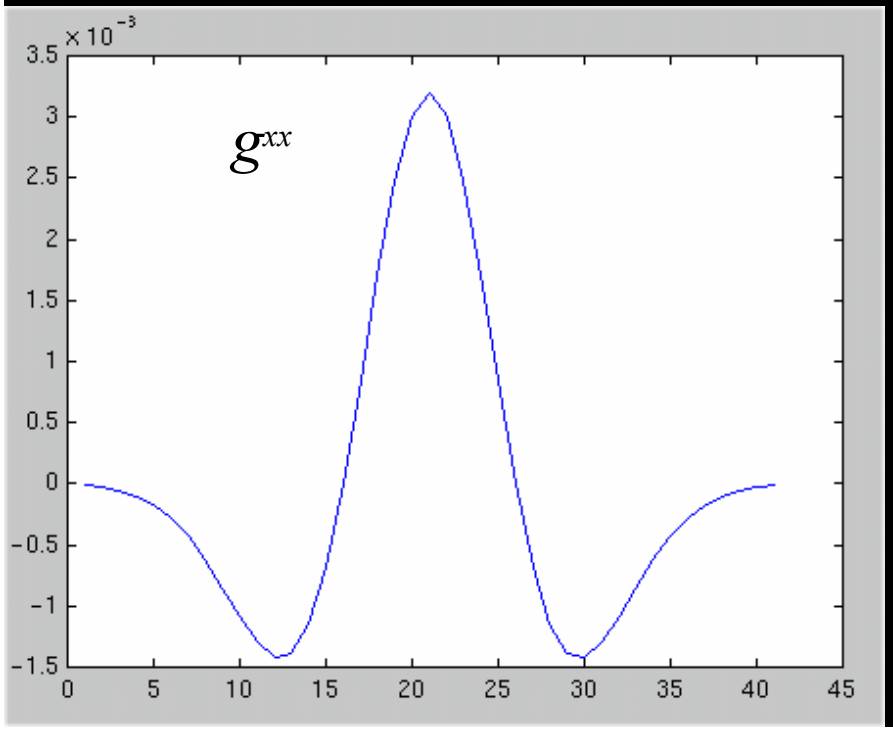




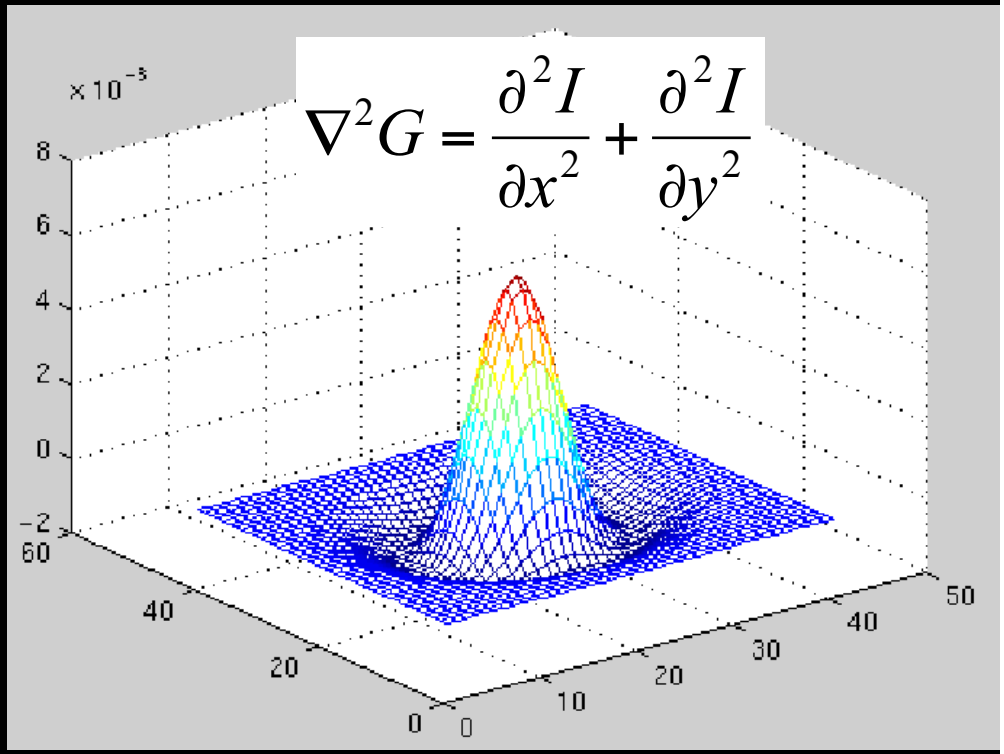
$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

Second derivatives: Laplacian



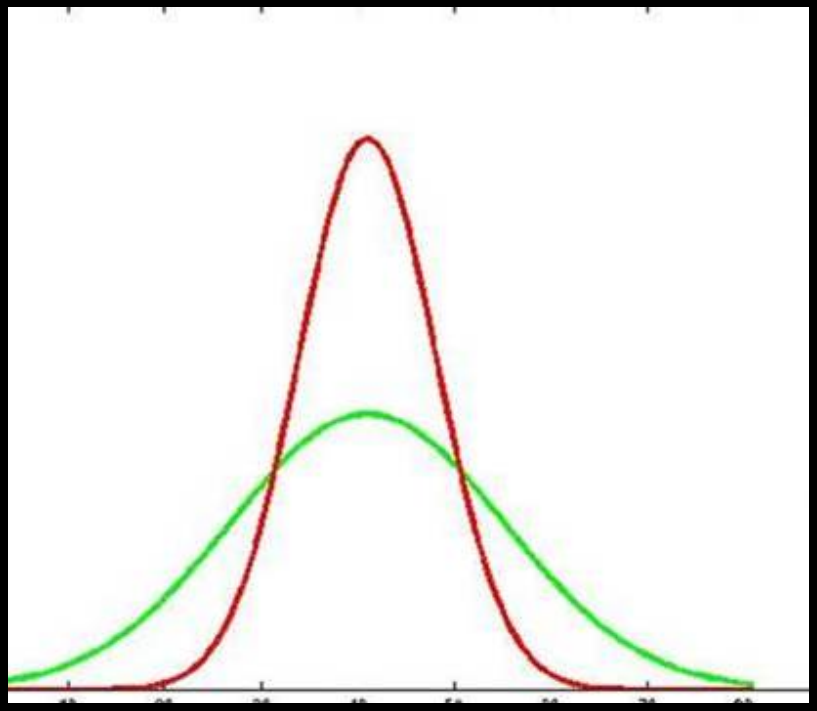


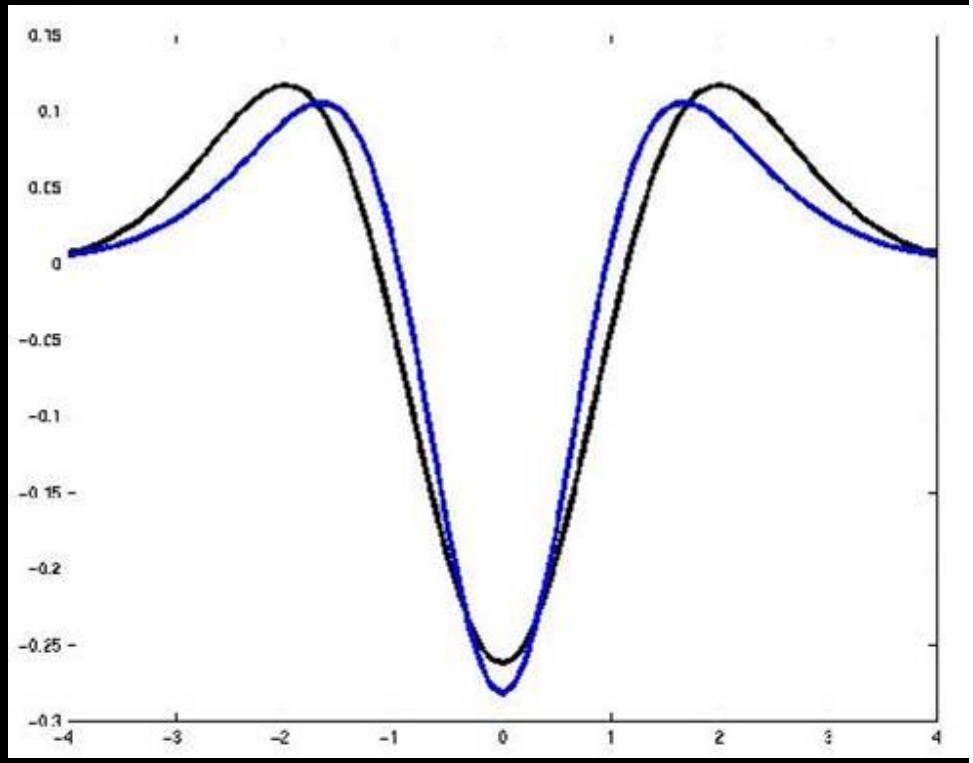
$$\nabla^2 G = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



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

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

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

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

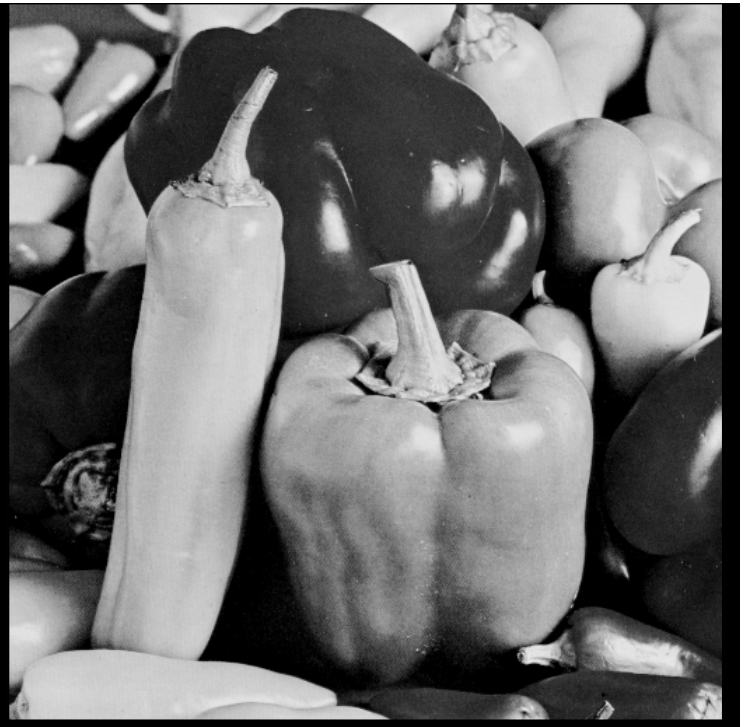
$$e^{-\frac{(a_1 x + b_1 y)^2}{2\sigma_1^2} - \frac{(a_2 x + b_2 y)^2}{2\sigma_2^2}}$$

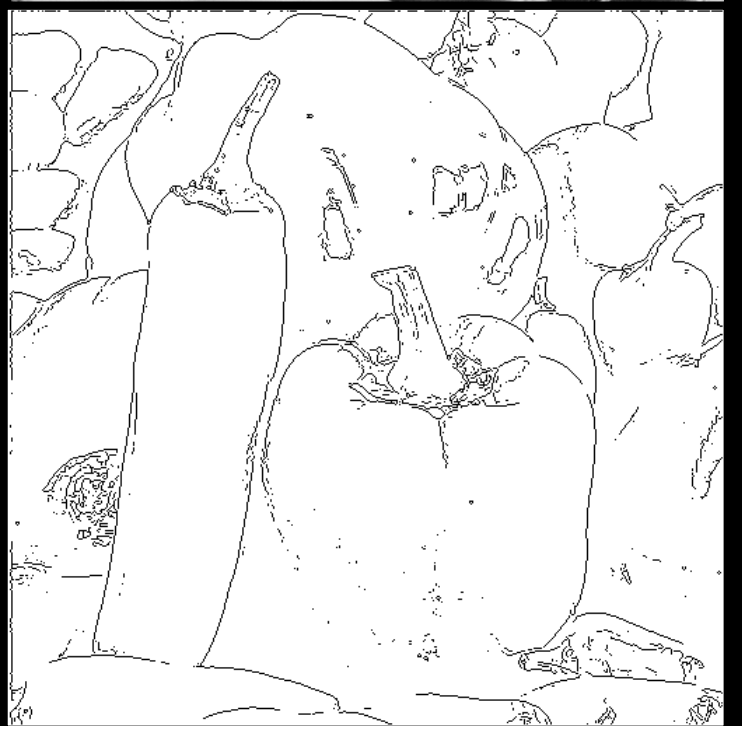
Smooth with different scales in orthogonal directions

Edge Detection

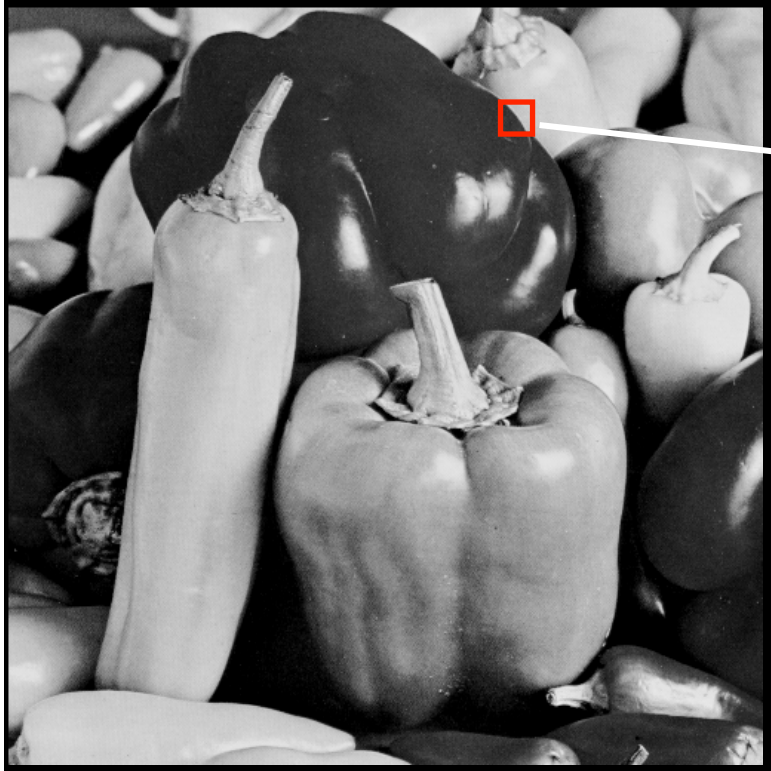
Edge Detection

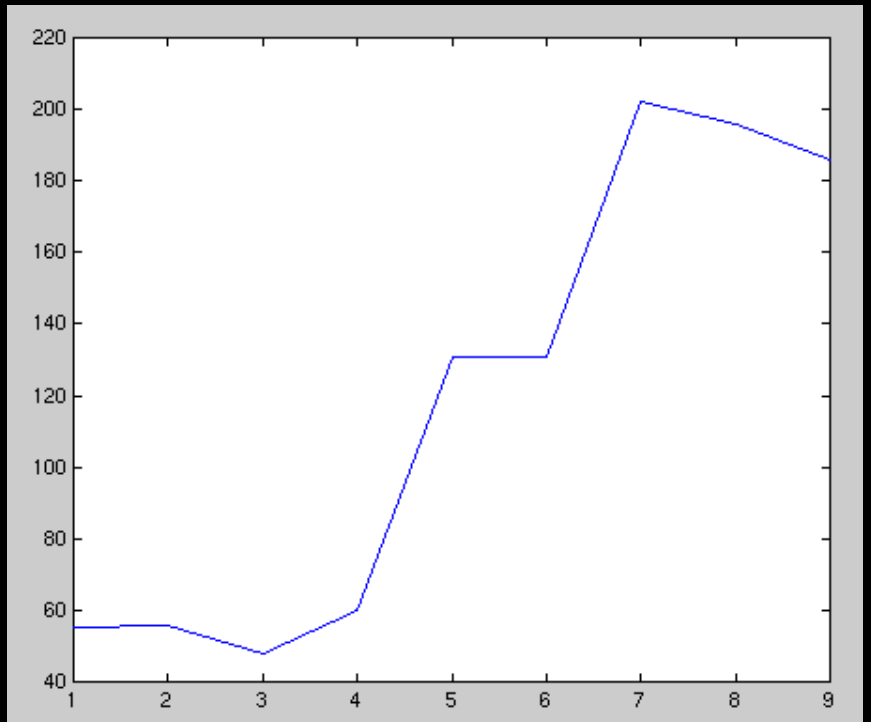
- Gradient operators
- Canny edge detectors
- Laplacian detectors





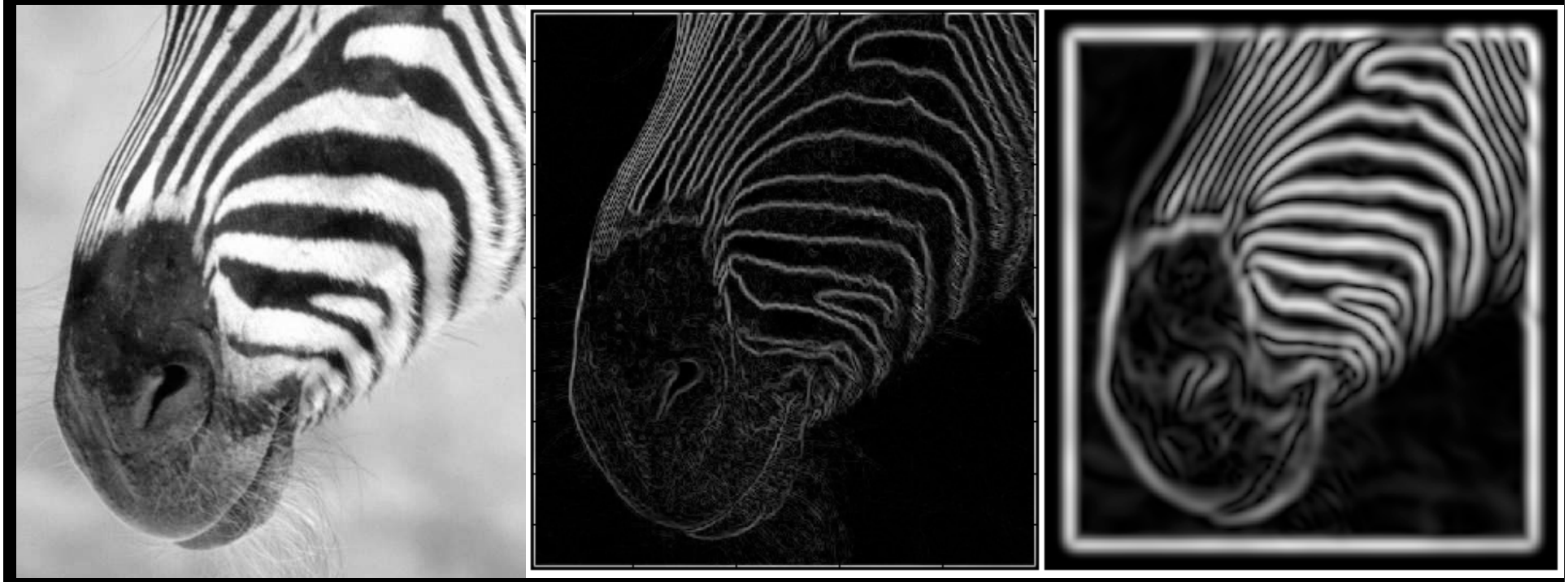
What is an edge?





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

Gradient-based edge detection



There are three major issues:

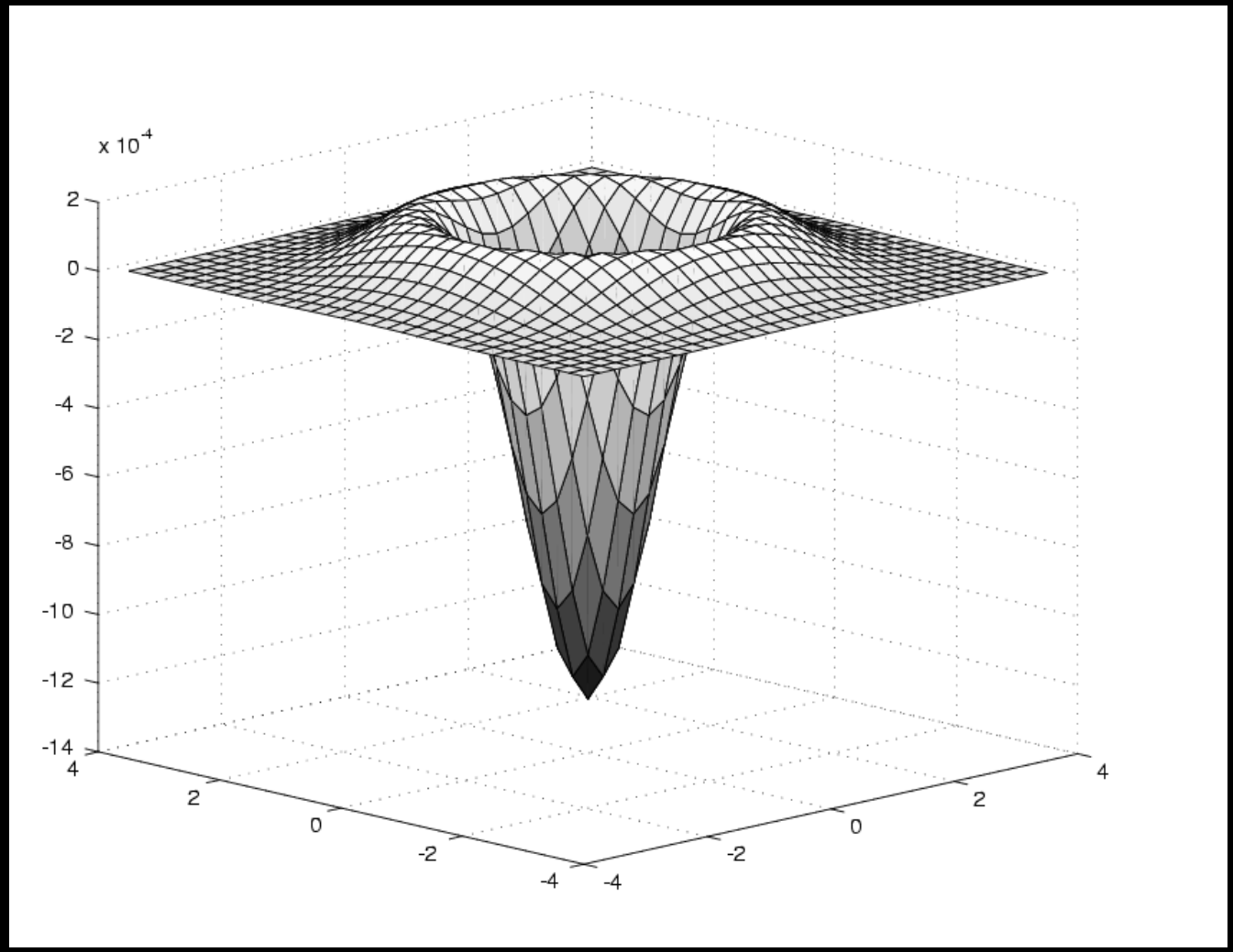
1) The gradient magnitudes at different scales are different; which one should we choose?
2) The gradient magnitude is large along thick trails; how do we identify the significant points?
3) How do we link the relevant points up into curves?

The Laplacian of Gaussian (Marr-Hildreth 80)

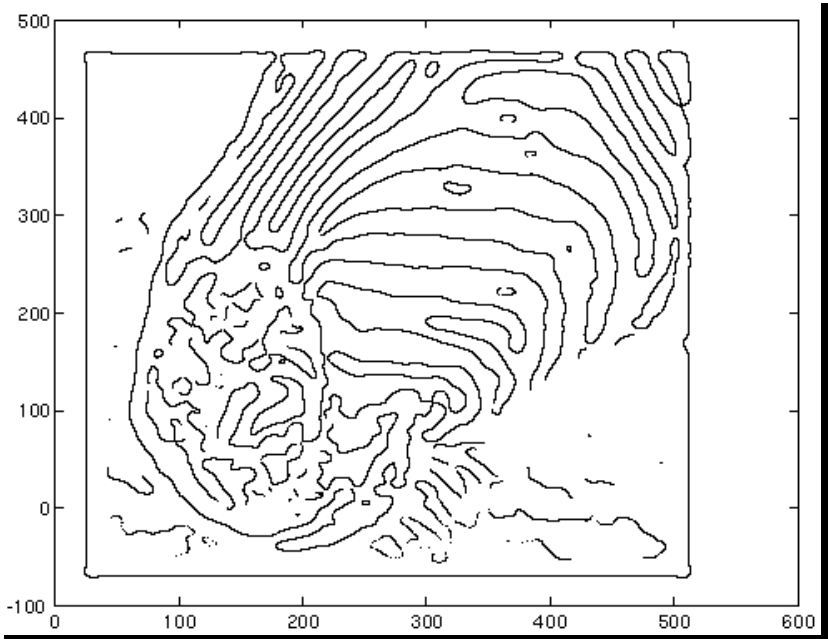
- Another way to detect an extremal first derivative is to look for a zero second derivative.
- Appropriate 2D analogy is rotation invariant:
  - the Laplacian

    $\nabla^2 f = \partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$

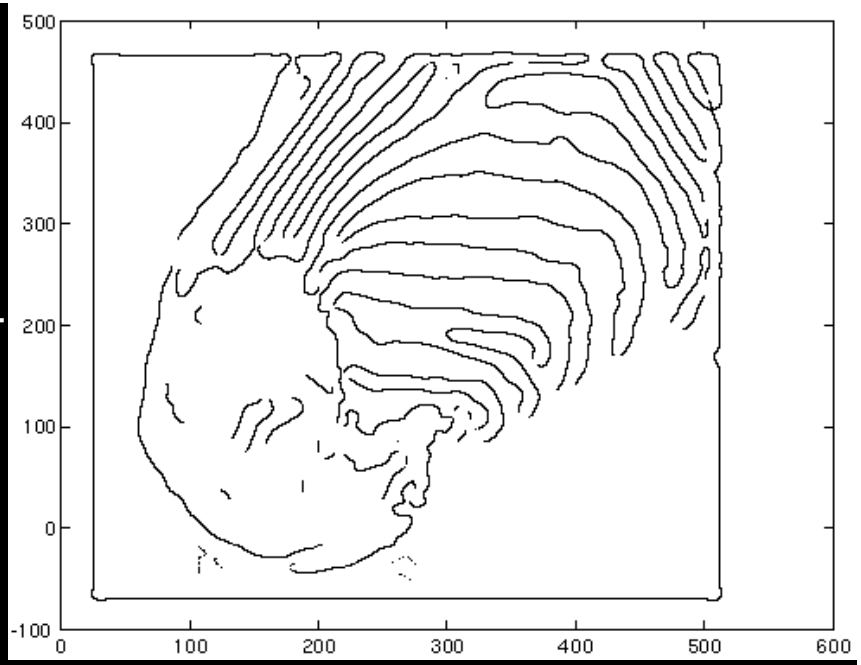
- Bad idea to apply a Laplacian without smoothing:
  - Smooth with Gaussian, apply Laplacian.
  - This is the same as filtering with a Laplacian of Gaussian filter.
- Now mark the zero points where there is a sufficiently large derivative, and enough contrast.



The Laplacian of a Gaussian



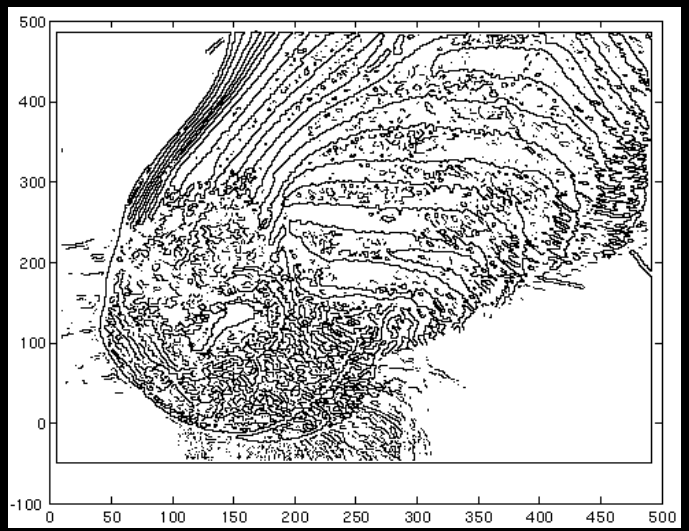
sigma=4



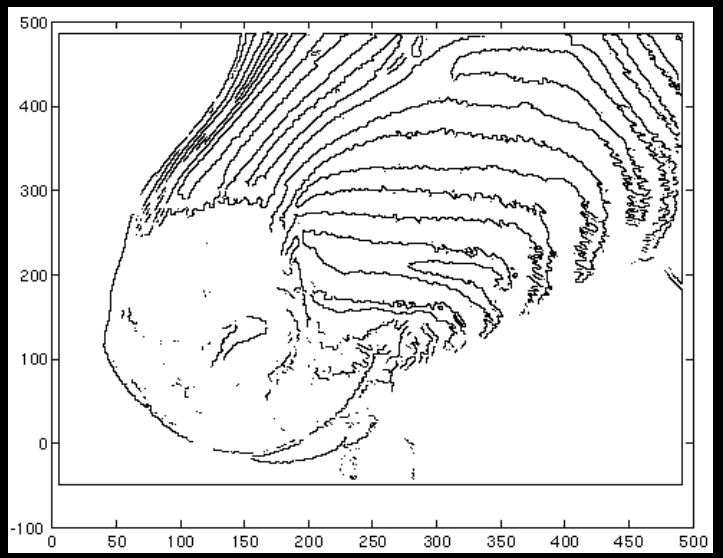
contrast=1

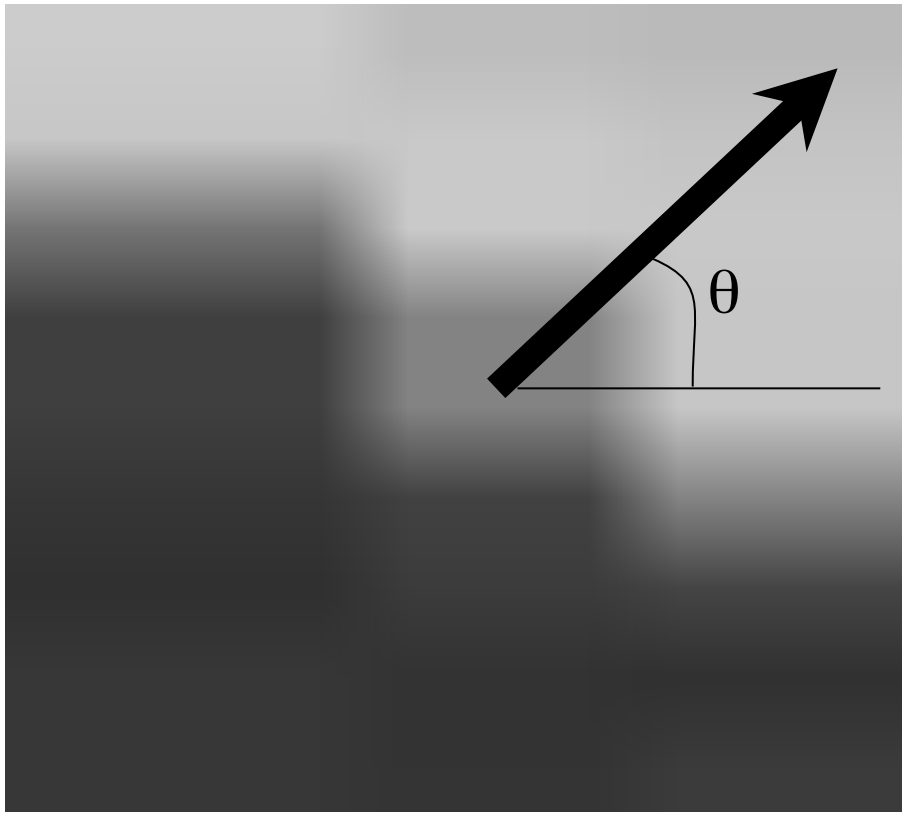
LOG zero crossings

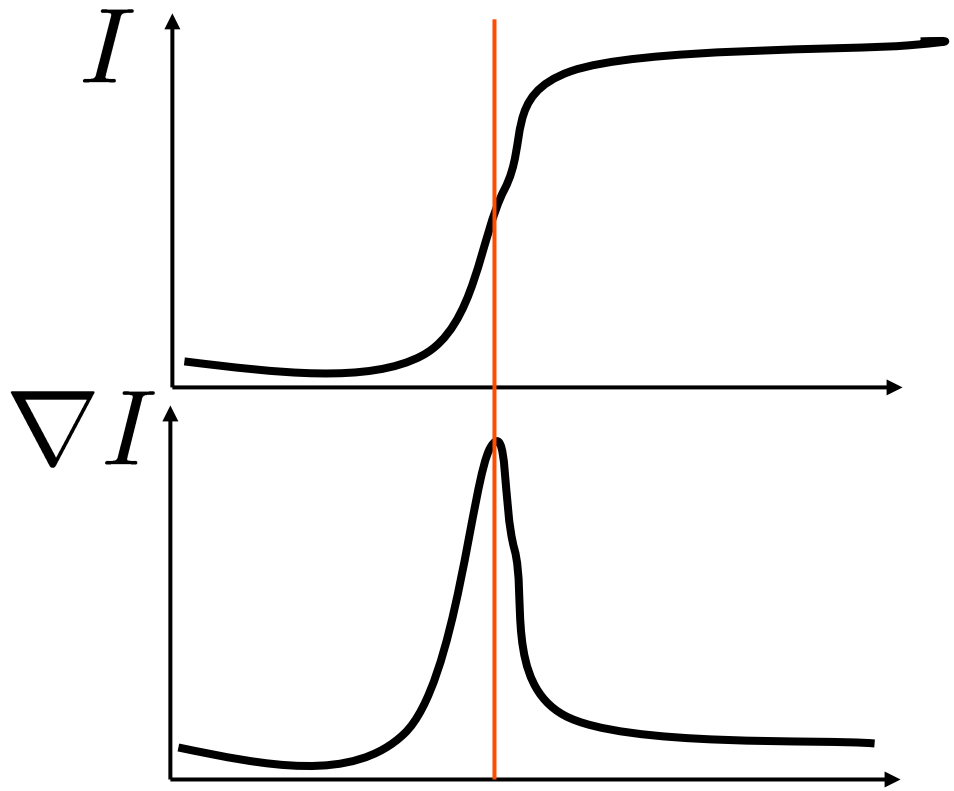
contrast=4



sigma=2



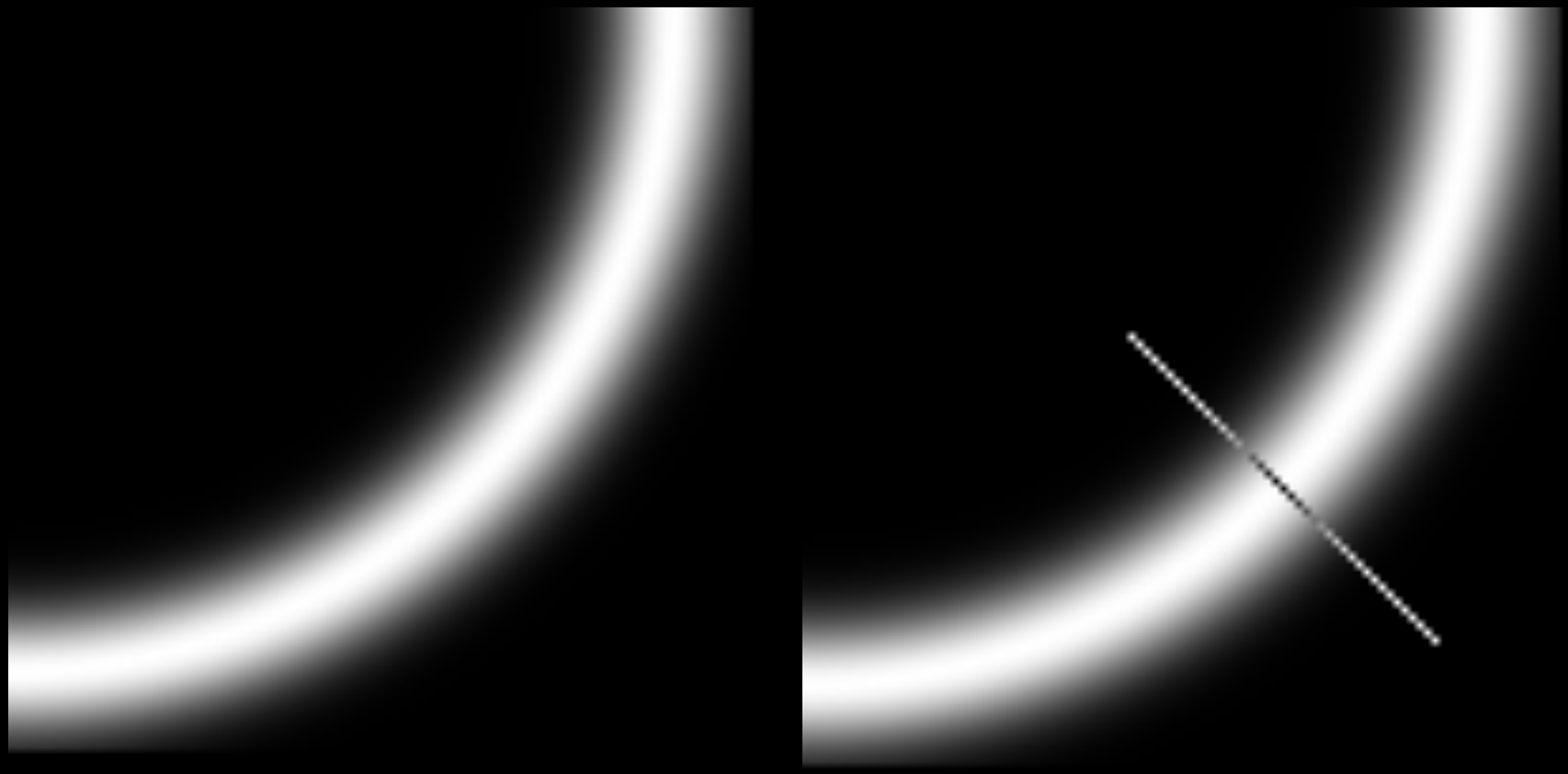




Edge pixels are at local maxima of gradient magnitude
Gradient computed by convolution with Gaussian derivatives
Gradient direction is always perpendicular to edge direction

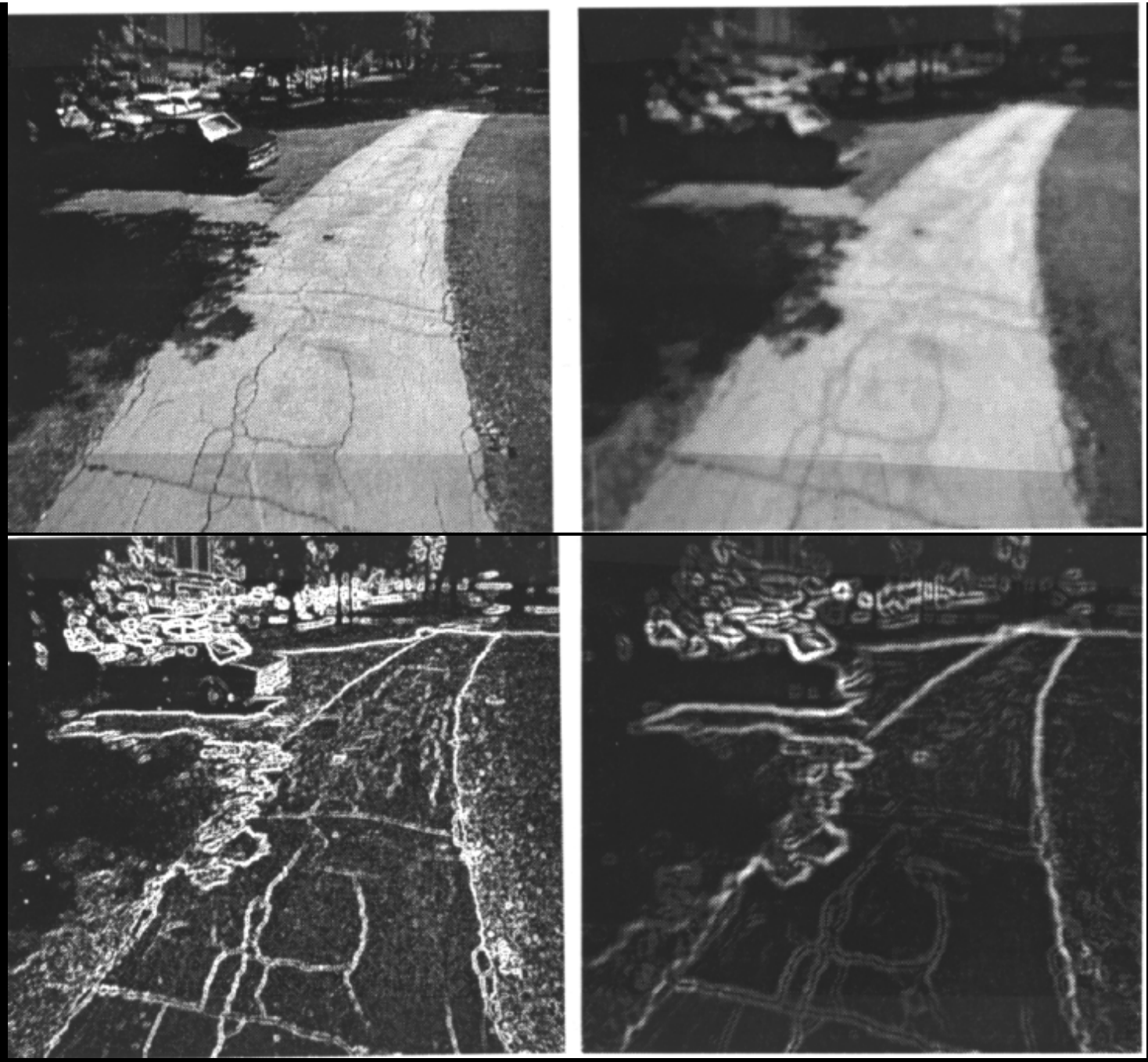
$$\frac{\partial I}{\partial x} = G_\sigma^x * I \qquad \frac{\partial I}{\partial y} = G_\sigma^y * I$$

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

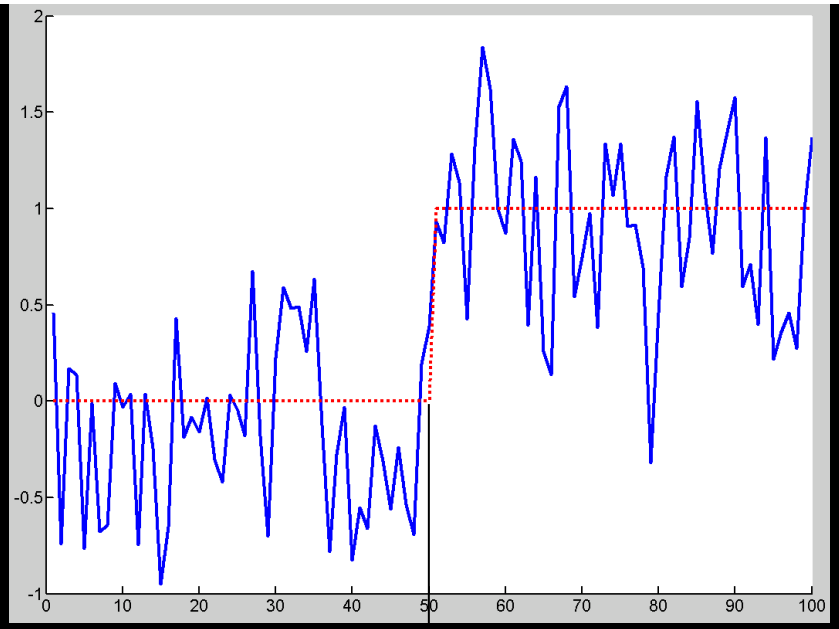


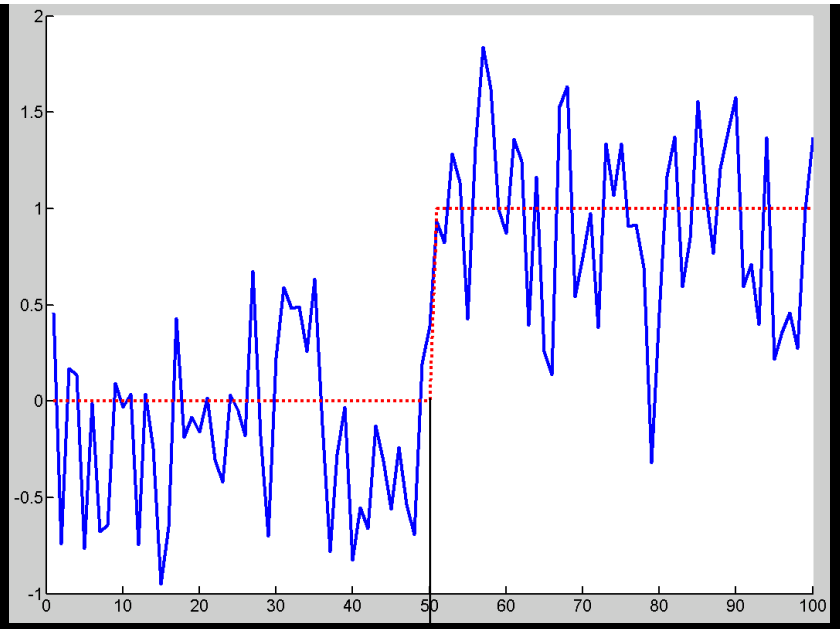
Gradient magnitude along an idealized curved edge.

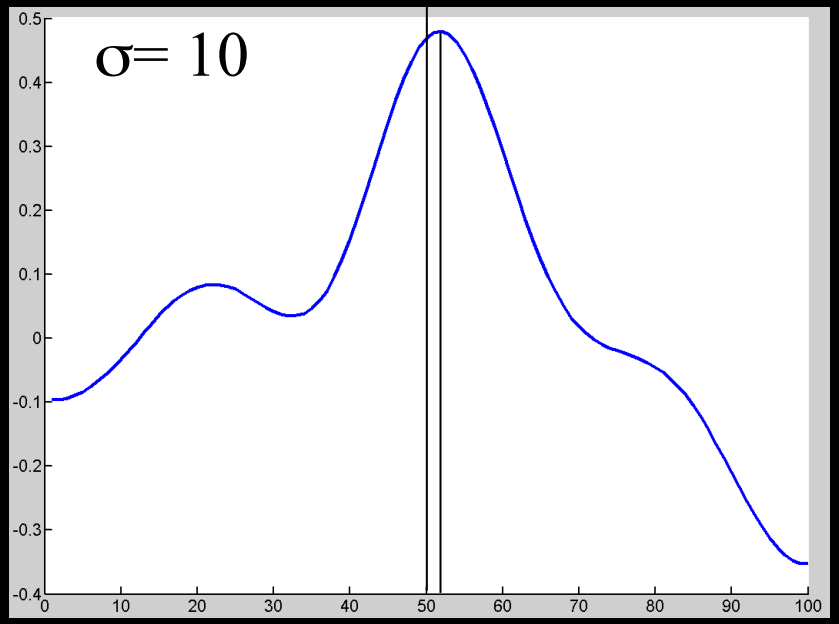
Curved edges are locally straight: The gradient is orthogonal to the edge direction.

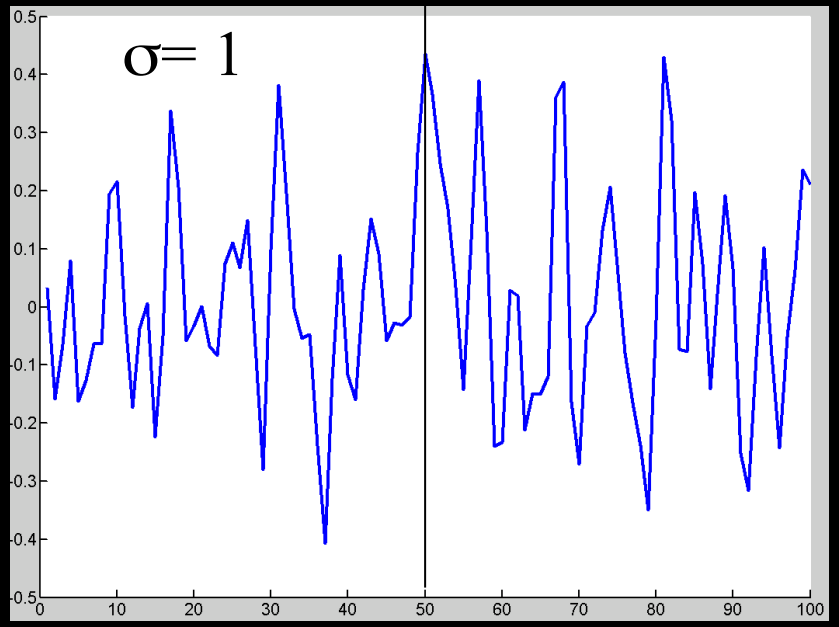


Small sigma Large sigma









Large σ → Good detection (high SNR)
Poor localization

Small σ → 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

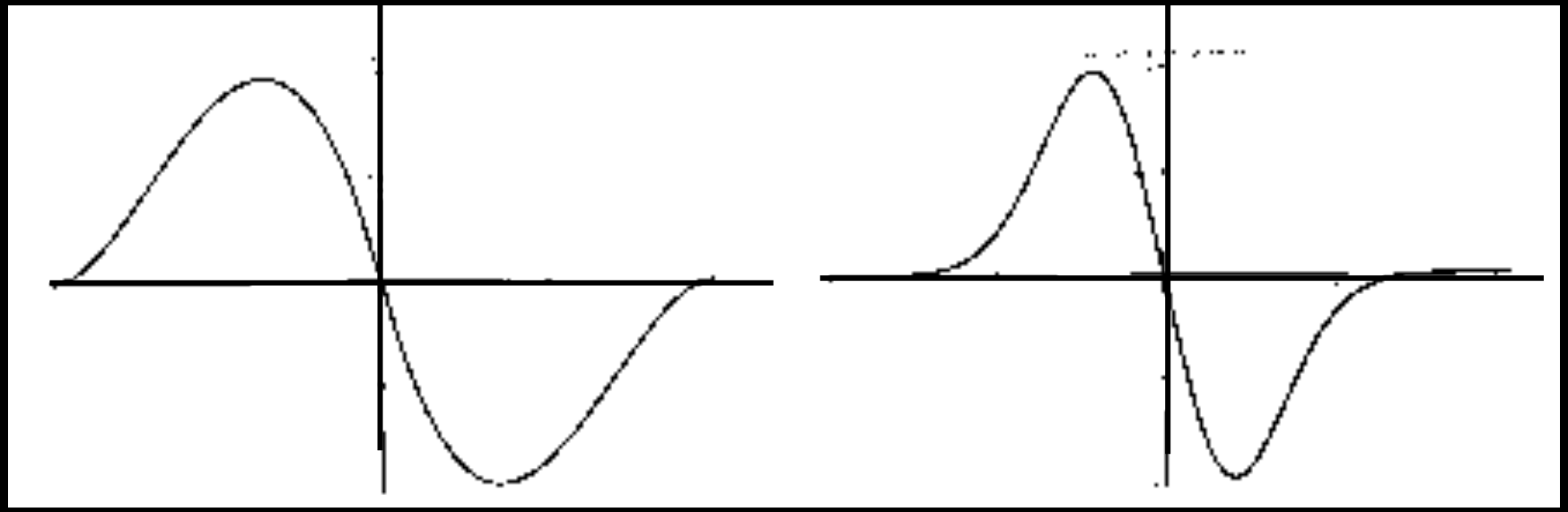
  $\Sigma(f)$ large if *f* produces good detection (high SNR)

- 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!

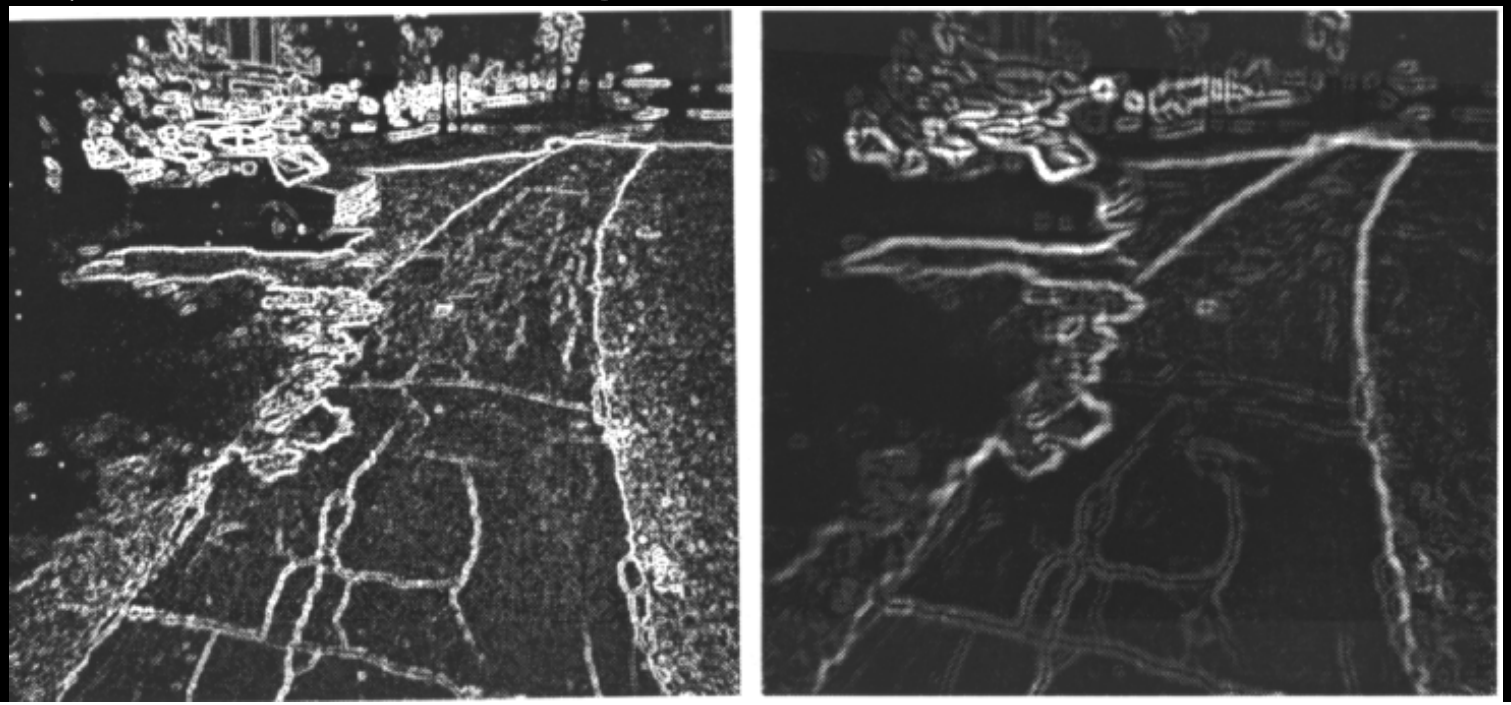


Canny

Derivative of Gaussian

Next Steps

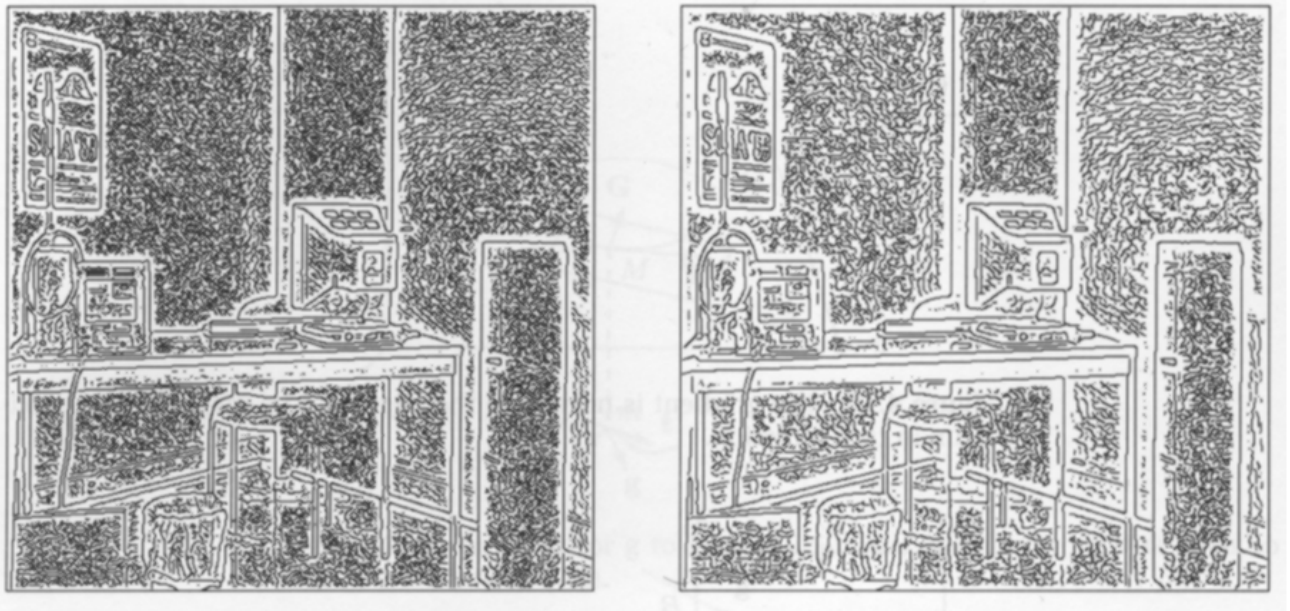
- The gradient magnitude enhances the edges but two problems remain:
  - What threshold should we use to retain only the "real" edges?
  - Even if we had a perfect threshold, we would still have poorly localized edges. How to extract optimally localize contours?
- Solution: Two standard tools:
  - Non-local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

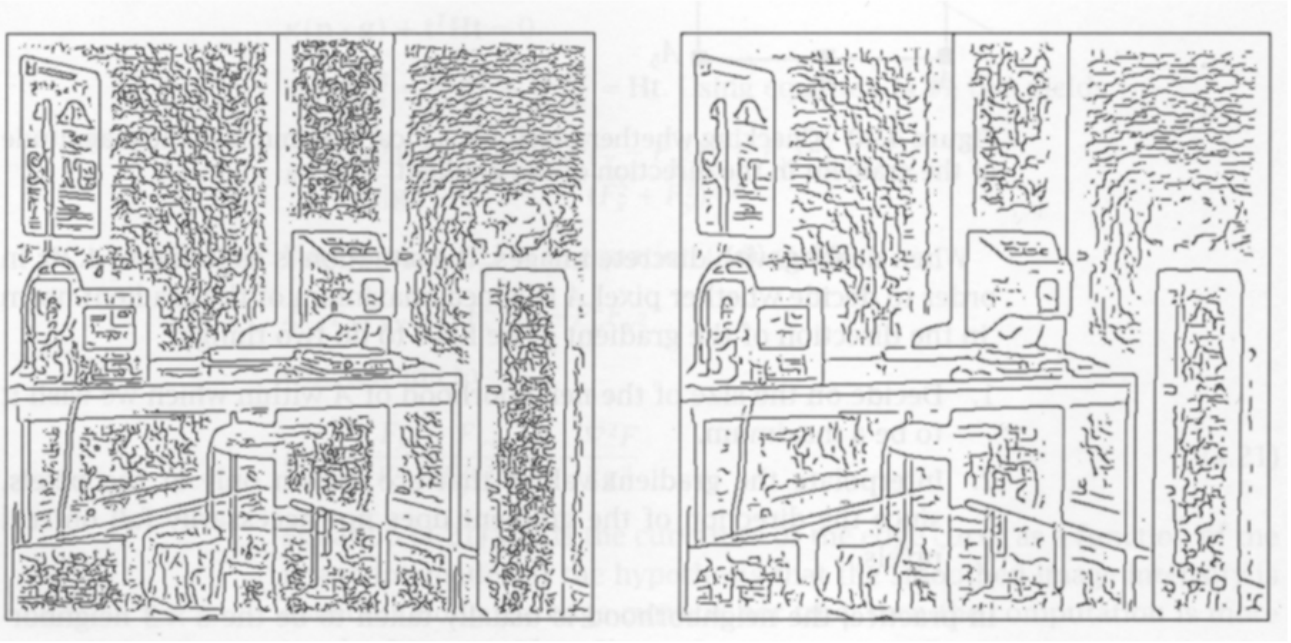


Different thresholds applied to gradient magnitude

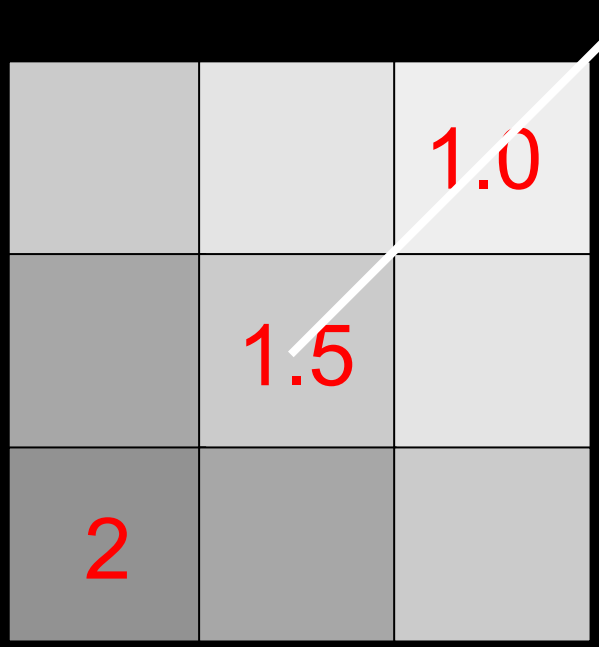
Input image

Different thresholds applied to gradient magnitude



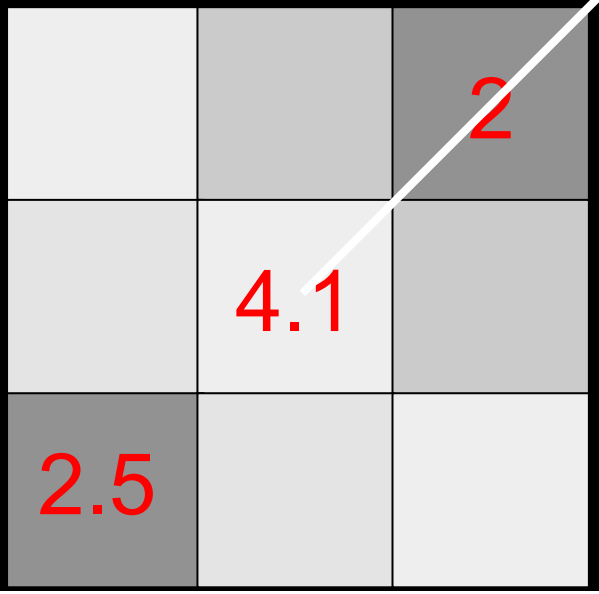


Non-Local Maxima Suppression



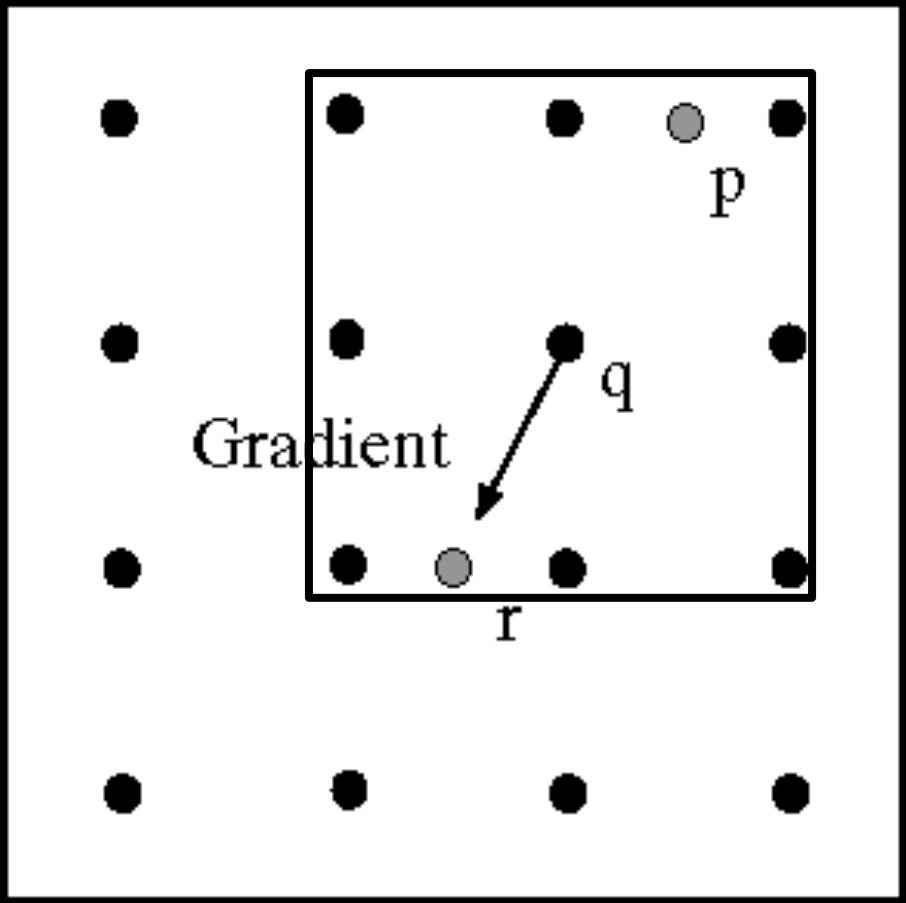
∇I

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)



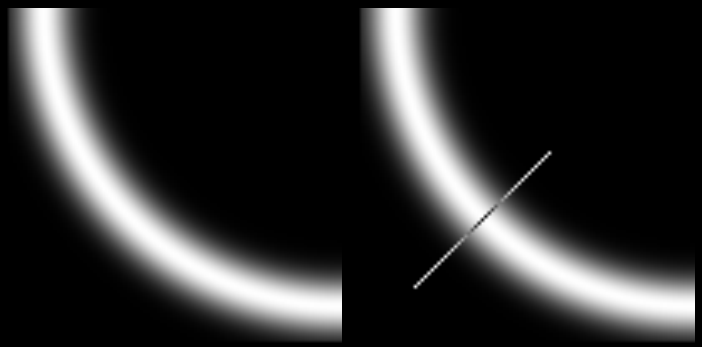
∇I

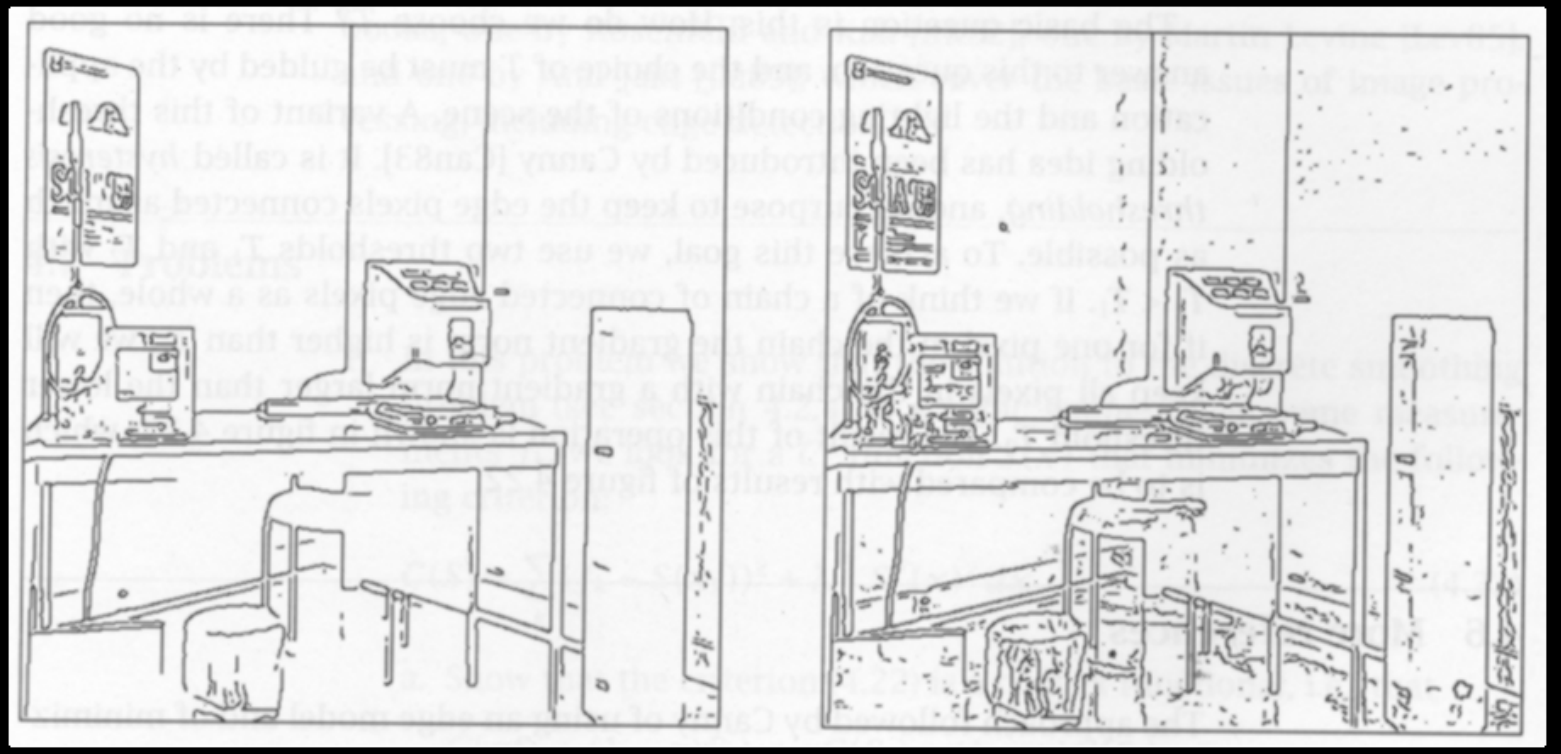
Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors in the direction of the gradient
→ Keep center pixel unchanged



Non-maximum suppression

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



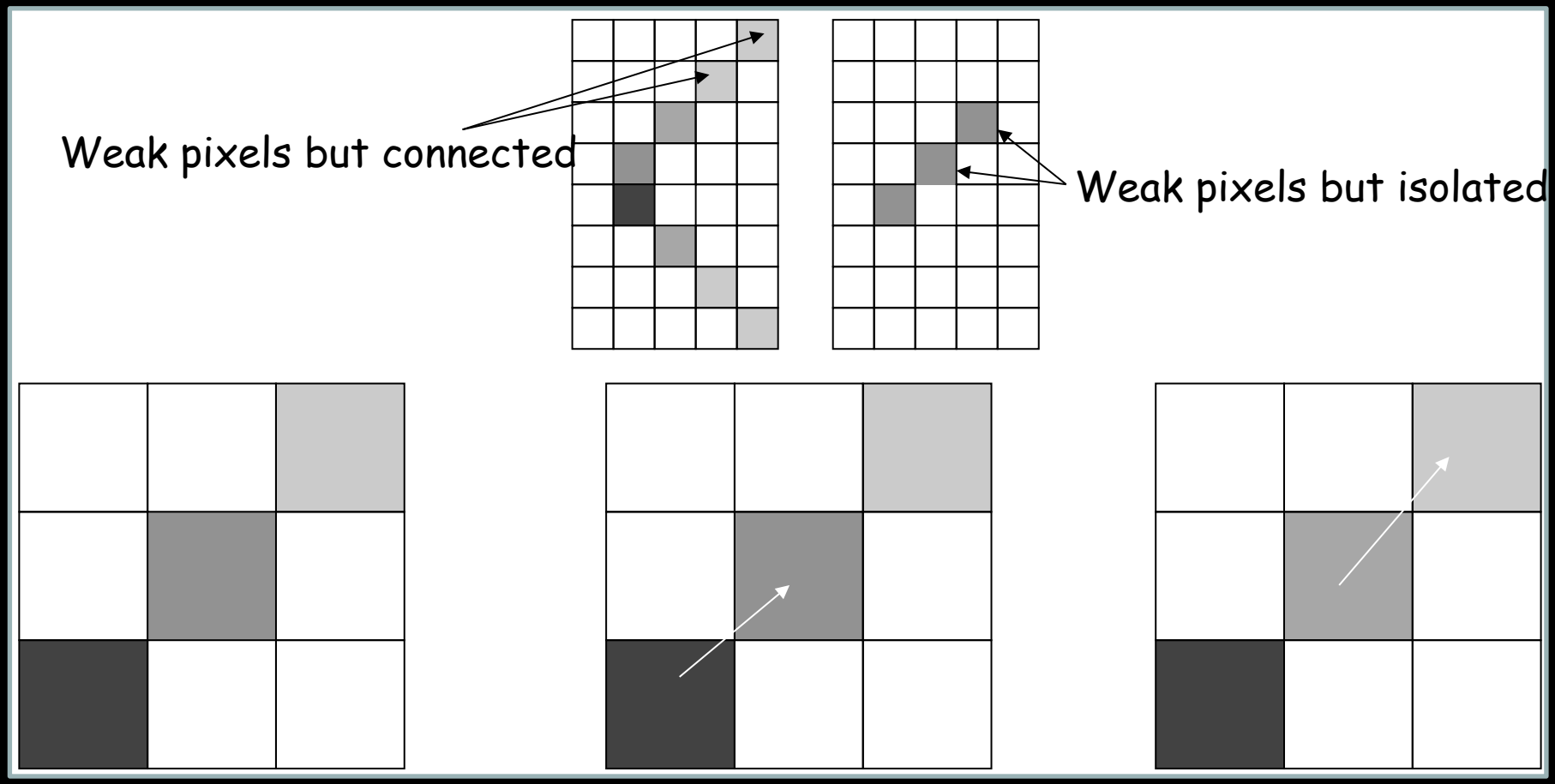


T = 15

T = 5

Two thresholds applied to gradient magnitude

Hysteresis Thresholding

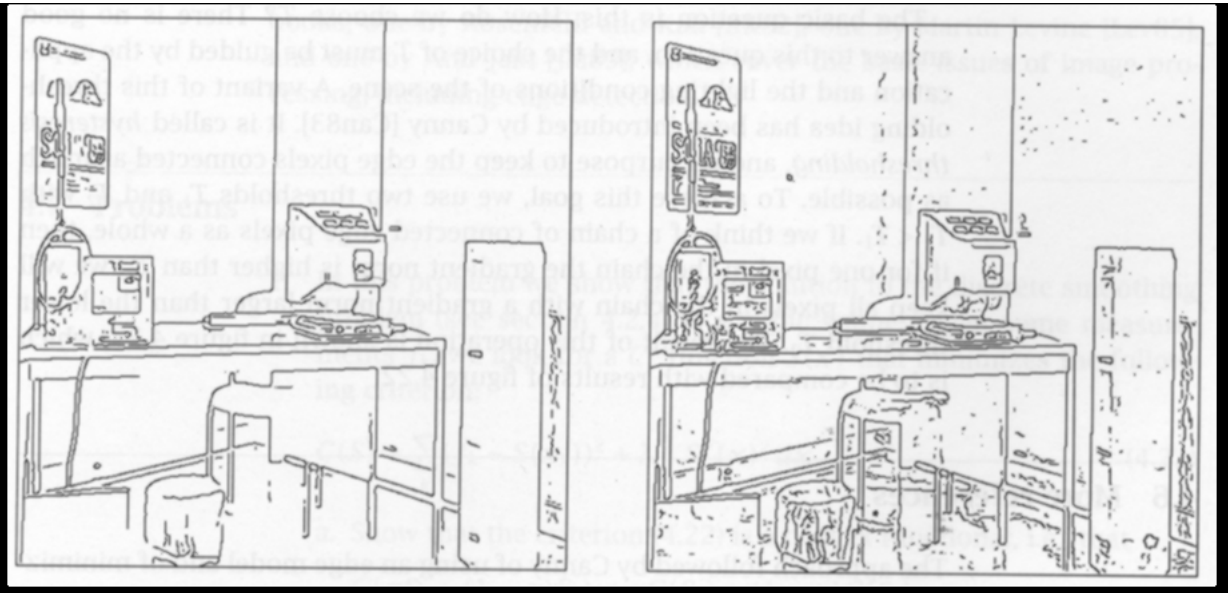


Very strong edge response. Let's start here

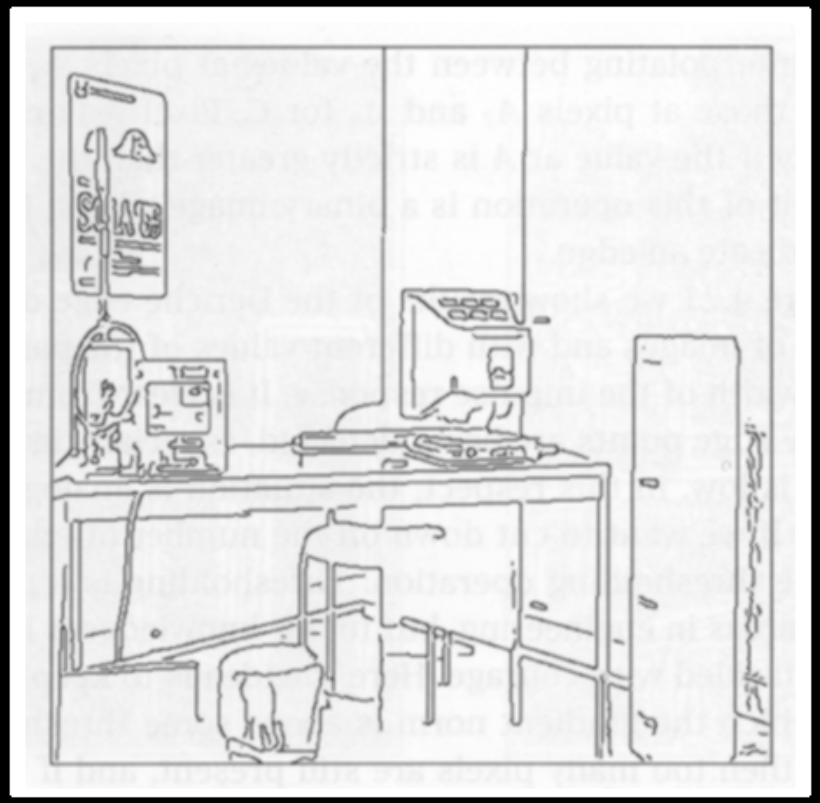
Weaker response but it is connected to a confirmed edge point. Let's keep it.

Continue…

T=15



T=5



Hysteresis
T_h=15 T_l = 5

Hysteresis thresholding

The Canny edge detector (1983)

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

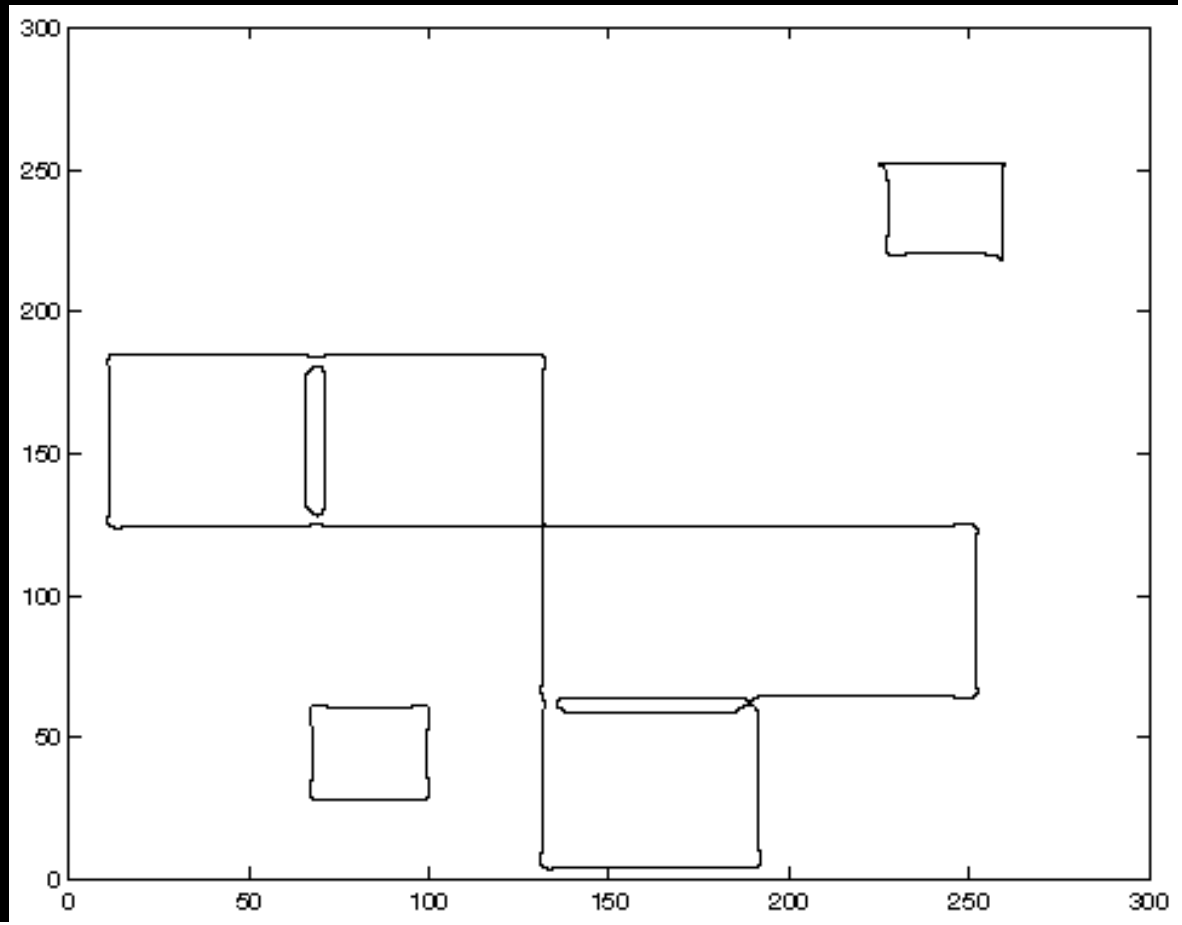
$$M(x, y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

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

We have unfortunate behaviour at corners



Summary

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