Cvision – 3
Image Pre-processing: Filtering & Enhancement

João Paulo Silva Cunha

(jcunha@det.ua.pt)

IEETA / Universidade de Aveiro

Outline

- Frequency Space
- Spatial Convolution
- Spatial filters
  - Gaussian; Oriented Gaussian; Mean; Median
- Frequency domain filtering
- Edge detection

Acknowledgements: Most of this course is based on the excellent courses offered by Prof. Shree Nayar at Columbia University, USA and by Prof. Srinivasa Narasimhan at CMU, USA. This was also based on Prof. Miguel Coimbra's slides. Please acknowledge the original source when reusing these slides for academic purposes.

Topic: Frequency Space

- Frequency Space
- Spatial Convolution
- Spatial filters
- Frequency domain filtering
- Edge detection

How does this apply to images?

- We have defined the Fourier Transform as

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-iux} dx$$

- But images are:
  - Discrete.
  - Two-dimensional.

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a computer sees

2D Discrete FT

- In a 2-variable case, the discrete FT pair is:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp[-j2\pi(ux/M + vy/N)]$$

For u=0,1,2,…,M-1 and v=0,1,2,…,N-1

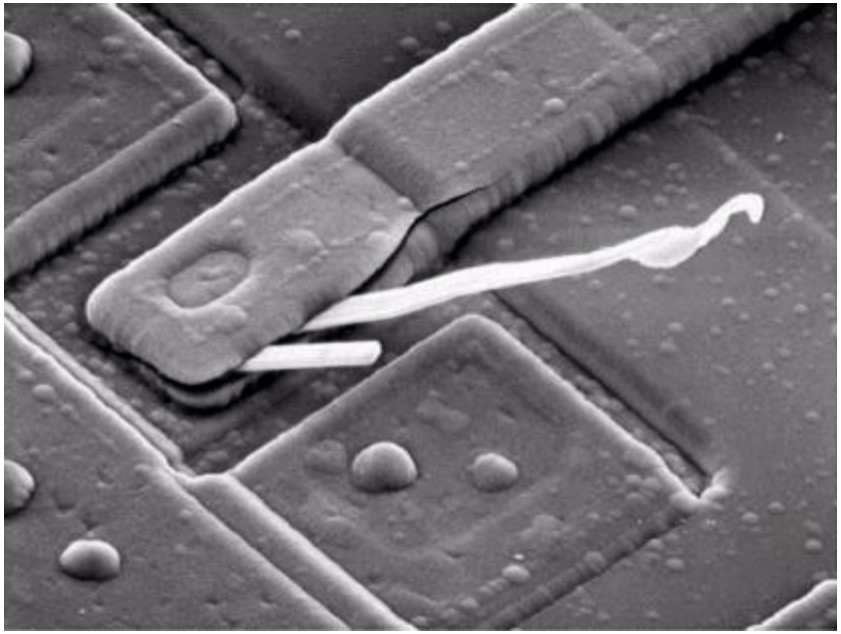
New matrix with the same size!

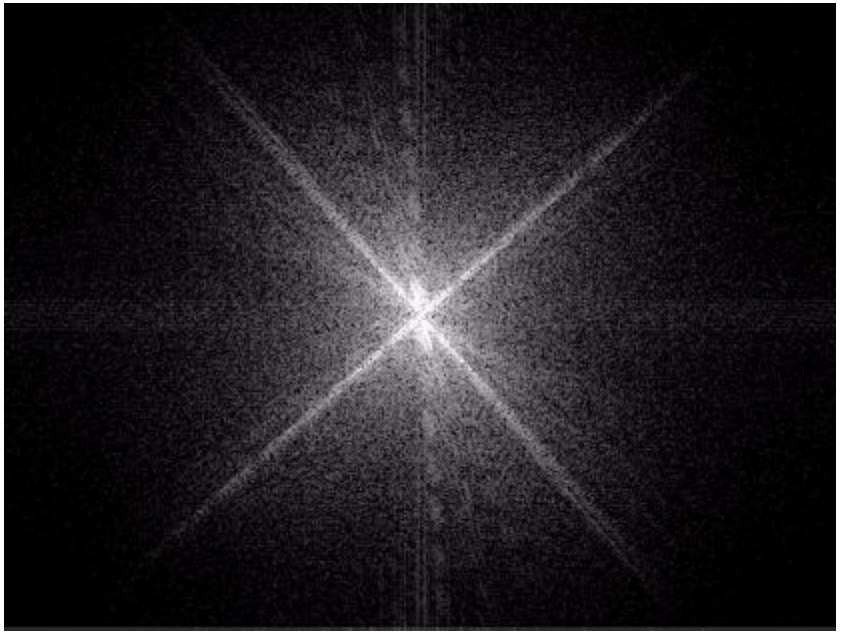
AND: $$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp[\, j2\pi(ux/M + vy/N)]$$

For x=0,1,2,…,M-1 and y=0,1,2,…,N-1

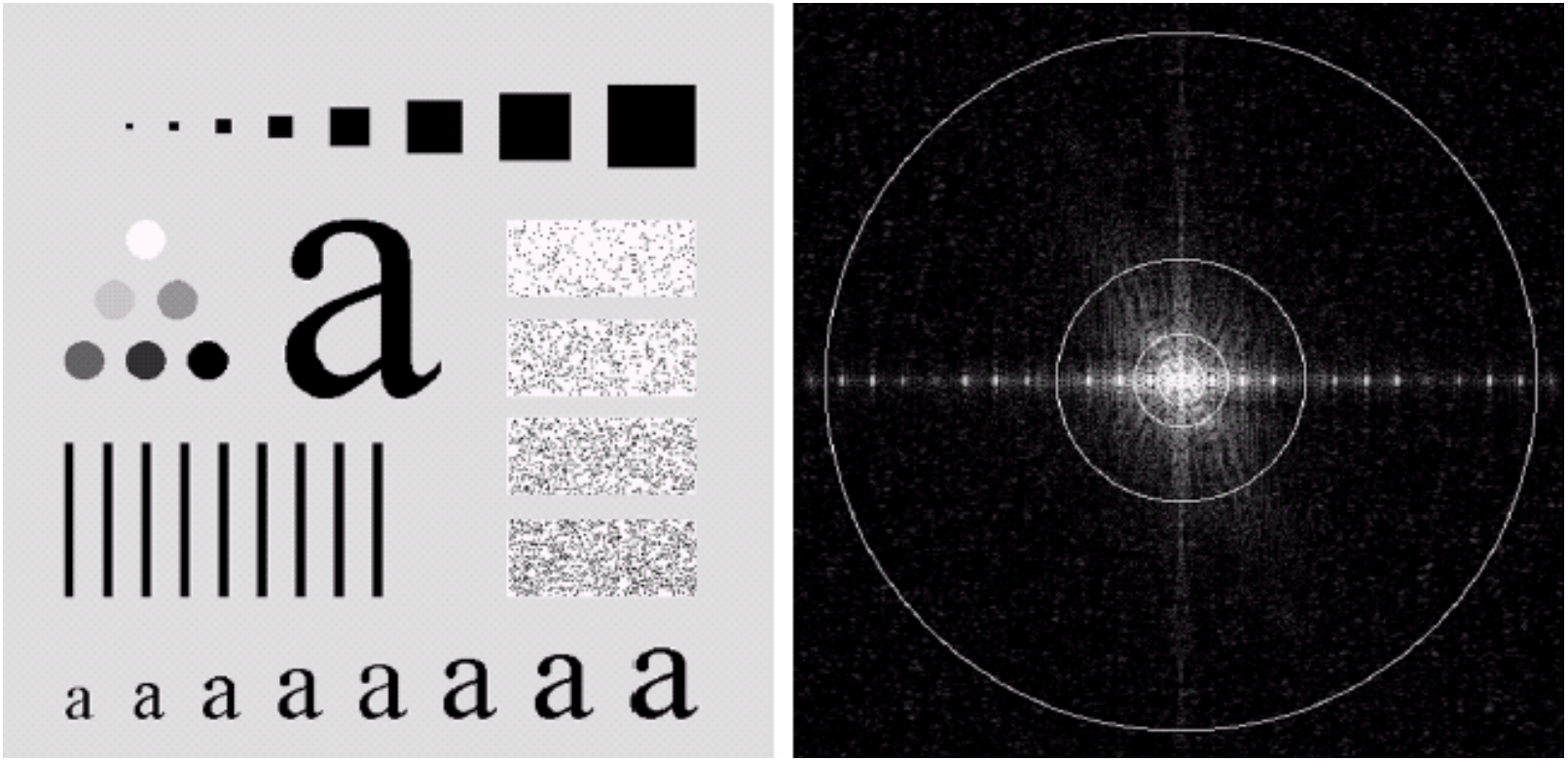
Frequency Space

- Image Space
  - f(x,y)
  - Intuitive
- Frequency Space
  - F(u,v)
  - What does this mean?





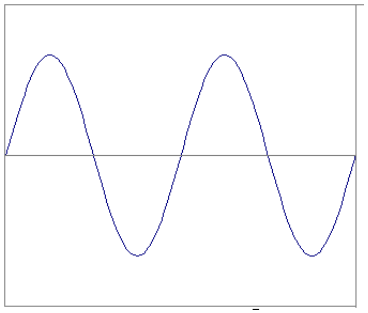
Power distribution



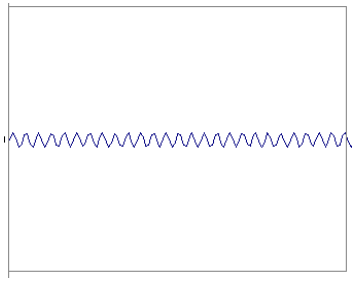
An image (500x500 pixels) and its Fourier spectrum. The super-imposed circles have radii values of 5, 15, 30, 80, and 230, which respectively enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power.

Power distribution

- Most power is in low frequencies.
- Means we are using more of this:

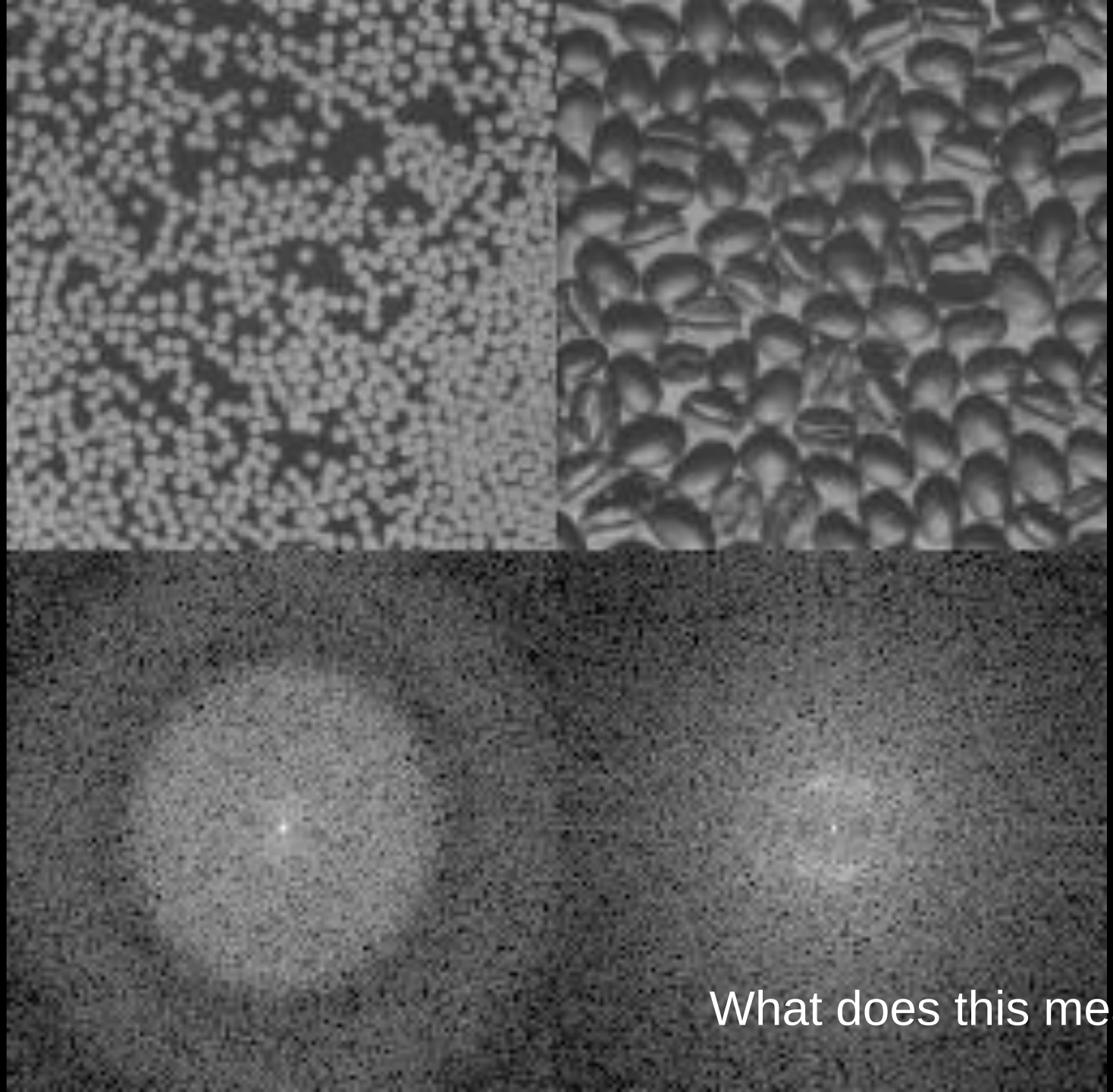


And less of this:



To represent our signal.

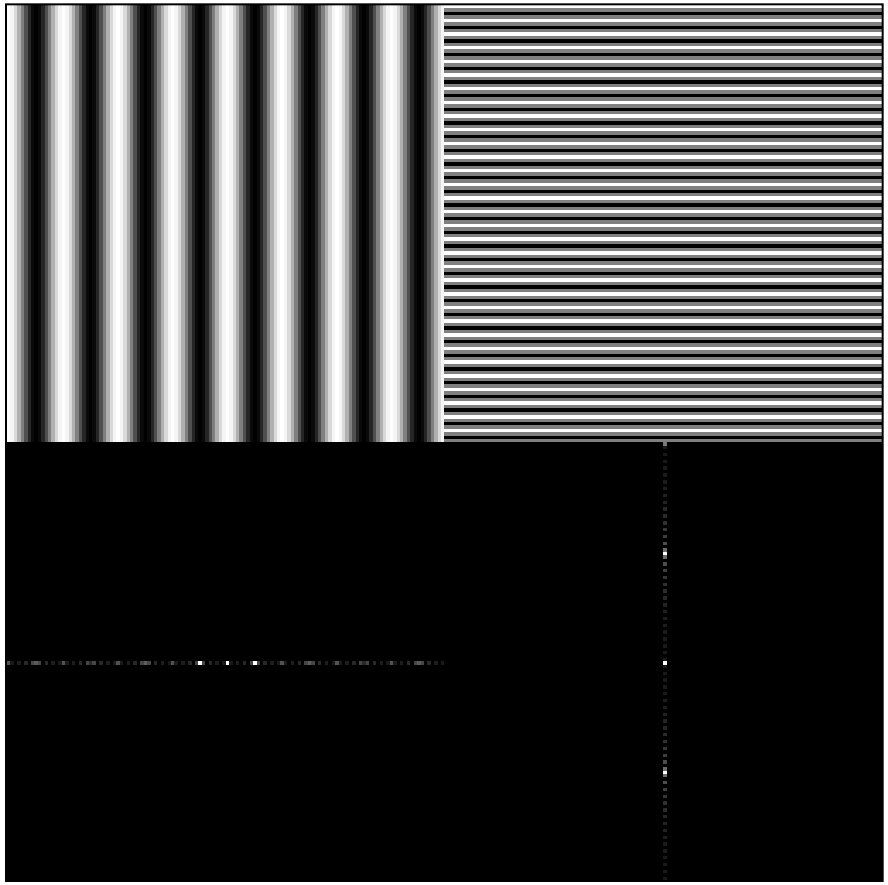
- Why?

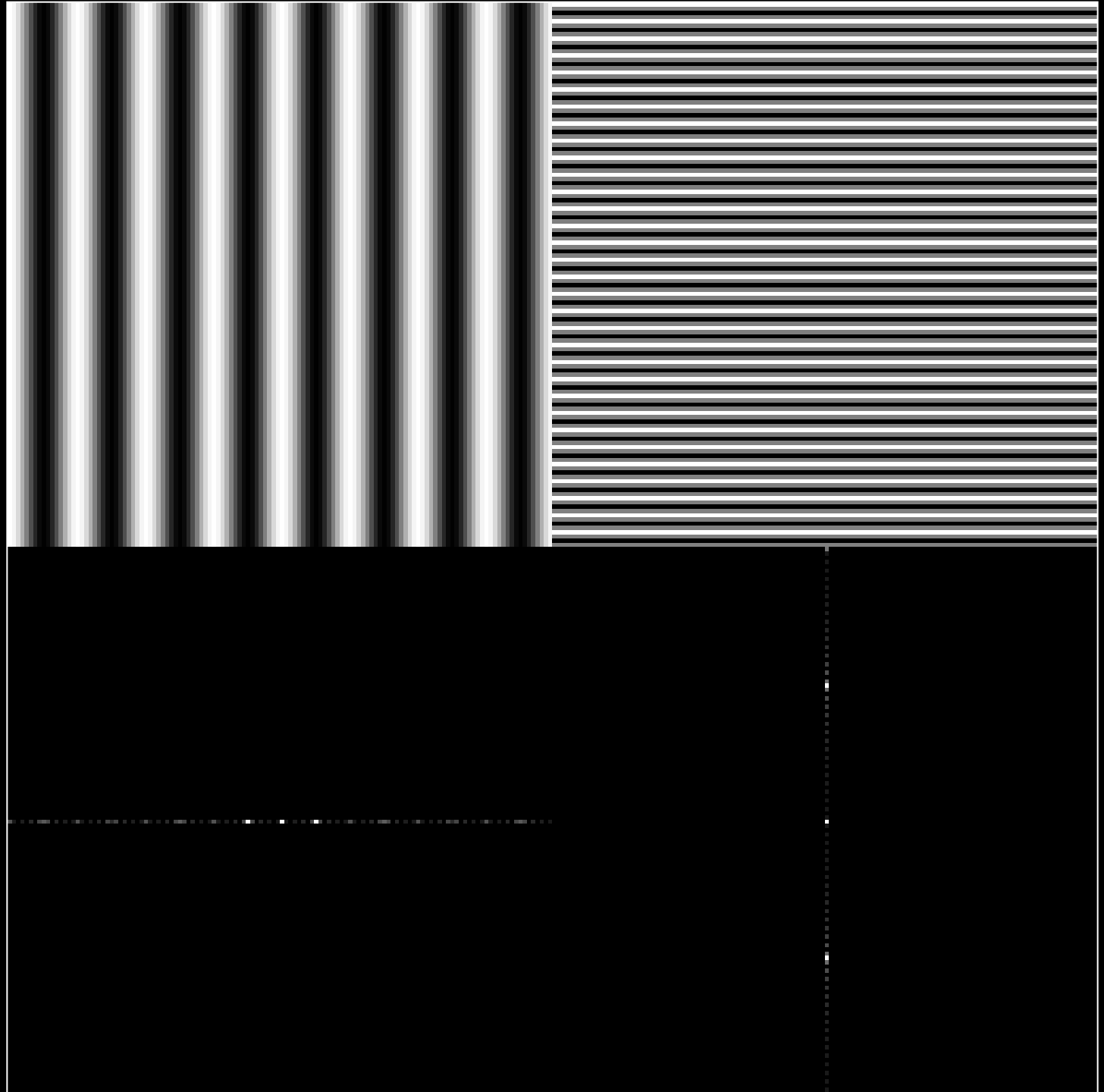


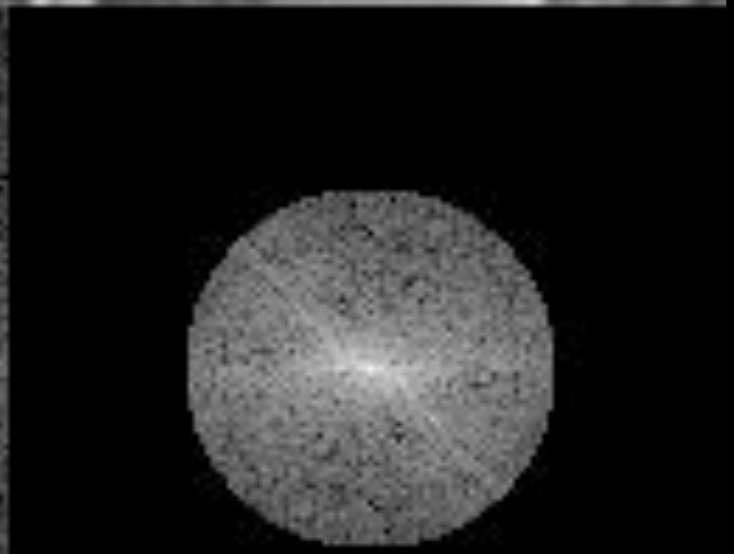
What does this mean??

Horizontal and Vertical Frequency

- Frequencies:
  - Horizontal frequencies correspond to horizontal gradients.
  - Vertical frequencies correspond to vertical gradients.
- What about diagonal lines?







If I discard high-frequencies, I get a blurred image...
Why?

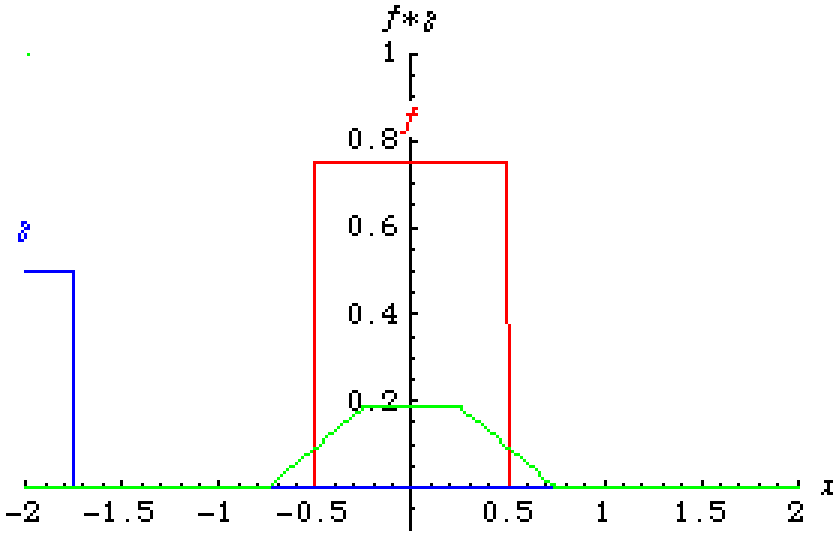
Why bother with FT?

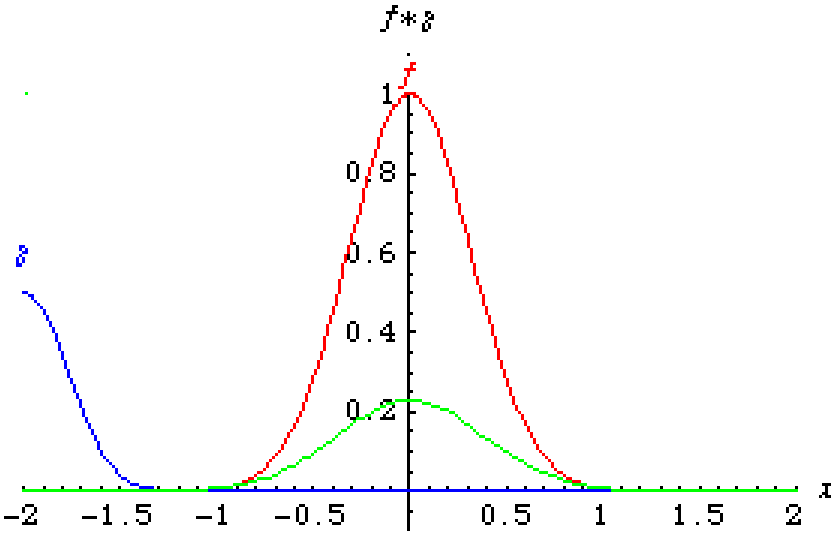
- Great for filtering.
- Great for compression.
- In some situations: Much faster than operating in the spatial domain.
- Convolutions are simple multiplications in Frequency space!
- ...

Topic: Spatial Convolution

- Frequency Space
- Spatial Convolution
- Spatial filters
- Frequency domain filtering
- Edge detection

Convolution





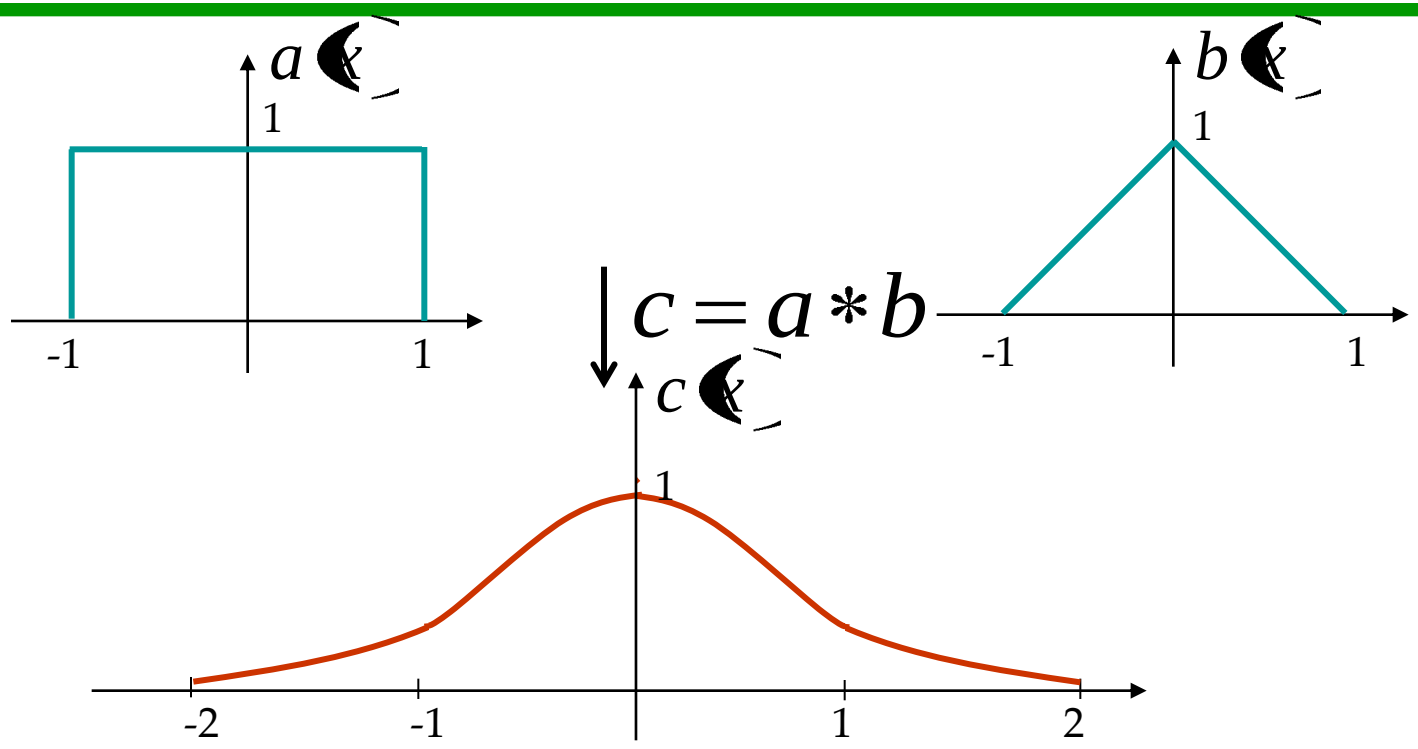
f

g

$f * g$

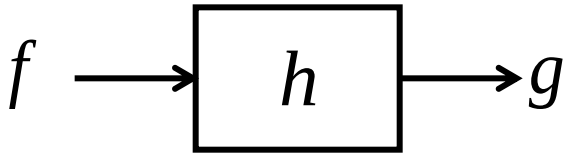
Eric Weinstein's Math World

Convolution



$$g(x) = \int_{-\infty}^{\infty} f(\tau)\, h(x-\tau)\, d\tau \qquad g = f * h$$

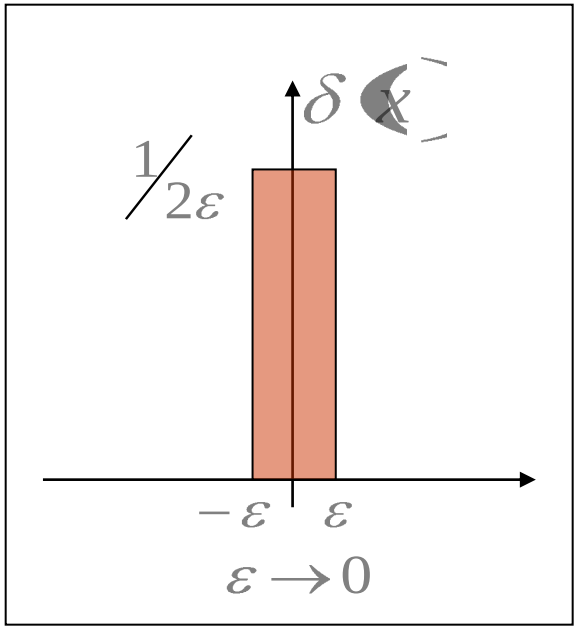
Convolution Kernel – Impulse Response



$$g = f * h$$

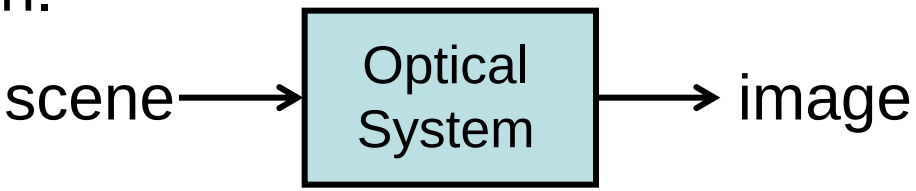
- What h will give us $g = f$?

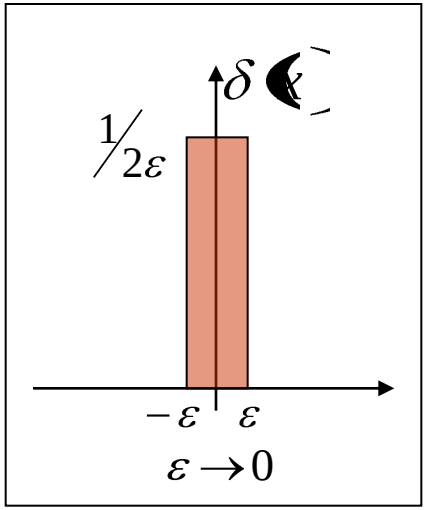
  Dirac Delta Function (Unit Impulse)



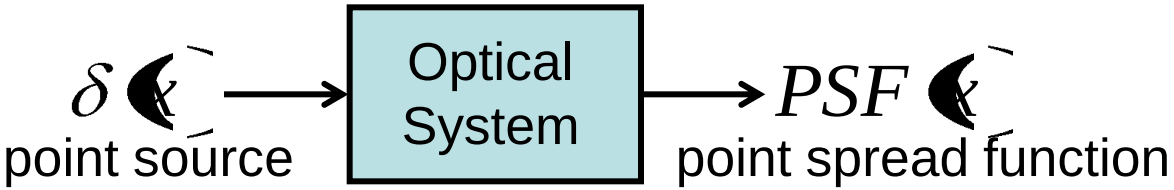
Point Spread Function

- Ideally, the optical system should be a Dirac delta function.

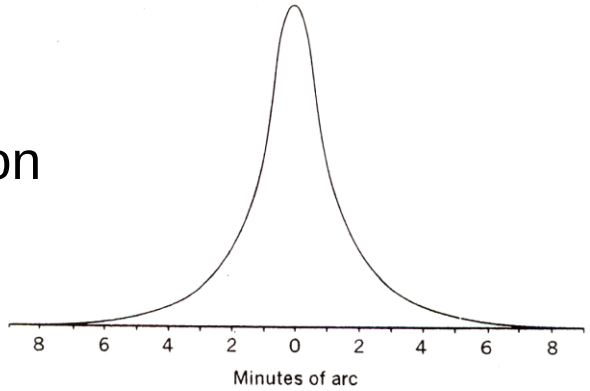




- However, optical systems are never ideal.



- Point spread function of Human Eyes.



Point Spread Function

normal vision

myopia

hyperopia

Images by Richmond Eye Associates

Properties of Convolution

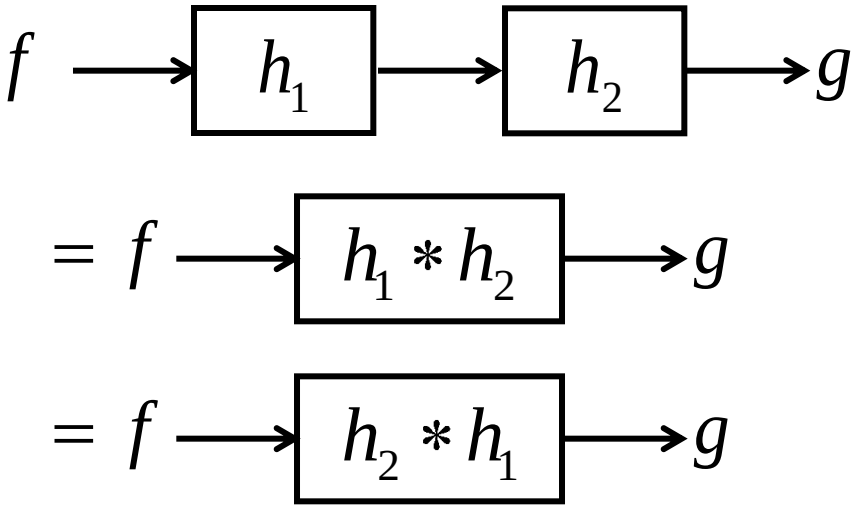
- Commutative

$$a * b = b * a$$

- Associative

$$(a * b) * c = a * (b * c)$$

- Cascade system



Fourier Transform and Convolution

Let $g = f * h$ Then $G(u) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi ux} dx$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) h(x-\tau) e^{-i2\pi ux} d\tau dx$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau \right] h(x-\tau) e^{-i2\pi u(x-\tau)} dx$$

$$= \int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau \right] \int_{-\infty}^{\infty} h(x') e^{-i2\pi ux'} dx' \qquad = F(u) H(u)$$

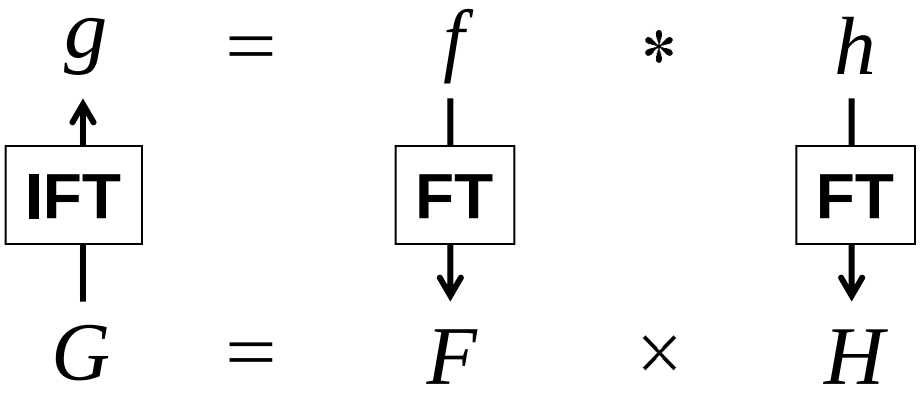
Convolution in spatial domain

$\Leftrightarrow$ Multiplication in frequency domain

Fourier Transform and Convolution

Spatial Domain *(x)*		Frequency Domain *(u)*
$g = f * h$	$\longleftrightarrow$	$G = FH$
$g = fh$	$\longleftrightarrow$	$G = F * H$

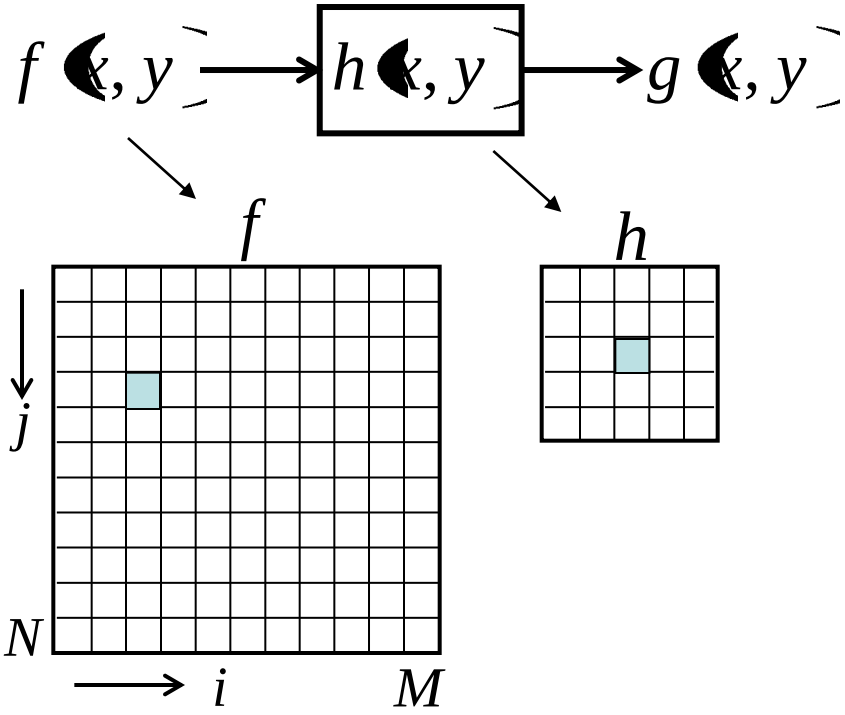
So, we can find *g(x)* by Fourier transform



Topic: Spatial filters

- Frequency Space
- Spatial Convolution
- Spatial filters
- Frequency domain filtering
- Edge detection

Images are Discrete and Finite



Convolution

$$g(i,j) = \sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)\,h(i-m, j-n)$$

Fourier Transform

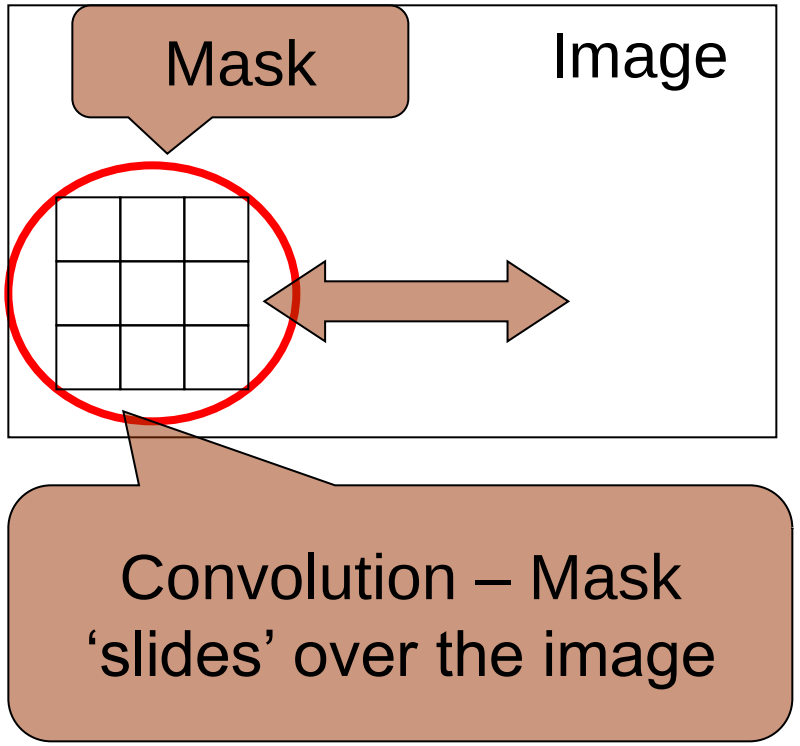
$$F(u,v) = \sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)\,e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

Inverse Fourier Transform

$$f(k,l) = \frac{1}{MN}\sum_{u=1}^{M}\sum_{v=1}^{N} F(u,v)\,e^{i2\pi\left(\frac{ku}{M}+\frac{lv}{N}\right)}$$

Spatial Mask

- Simple way to process an image.
- Mask defines the processing function.
- Represents a spatial convolution that corresponds to a multiplication in frequency domain.



Example (1 pixel)

- Each mask position has weight **w**.
- The result of the operation for each pixel is given by:

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

Mask

2	2	2
4	4	4
4	5	6

Image

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

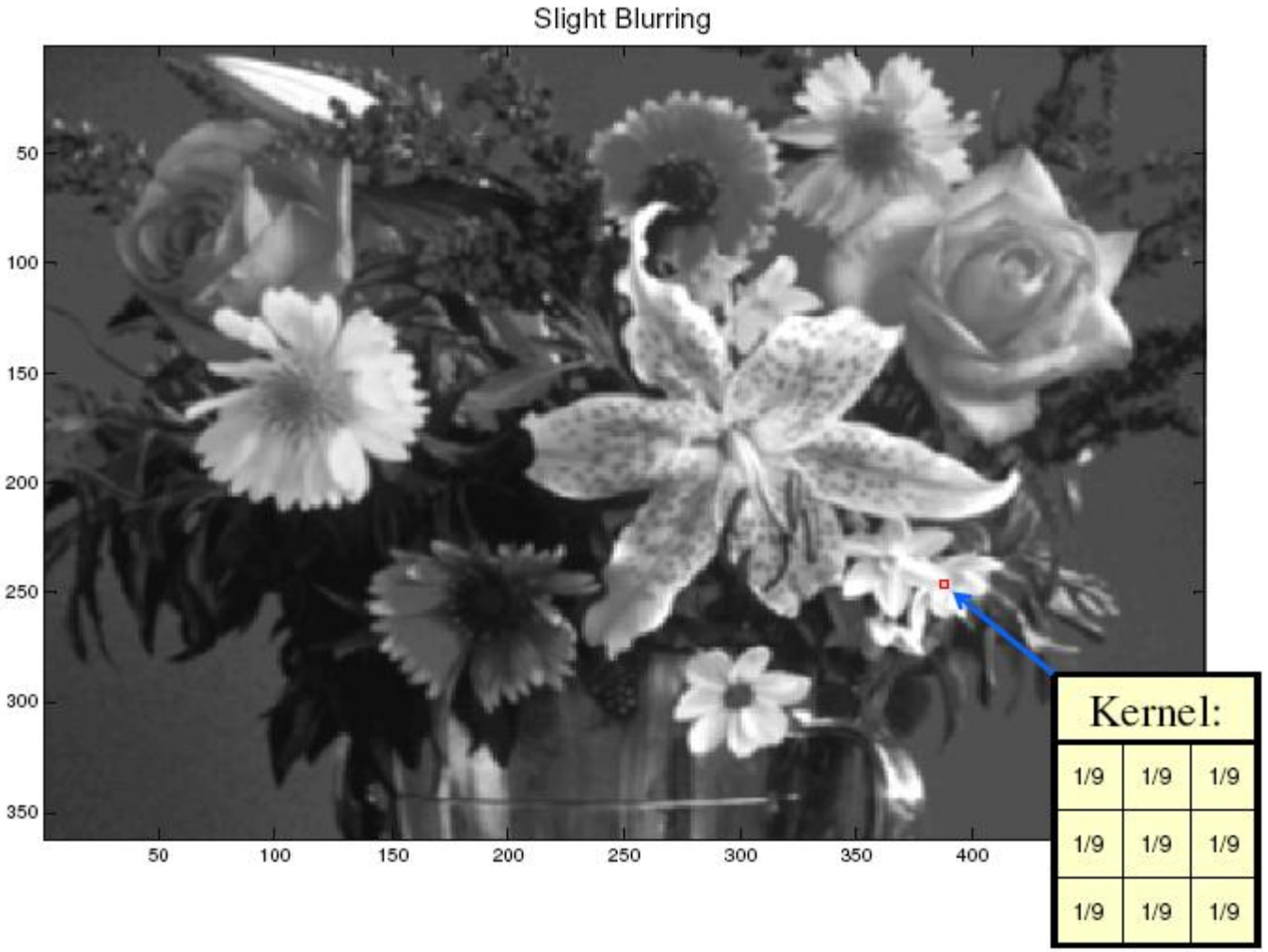
=1*2+2*2+1*2+…
=8+0-20
=-12

Definitions

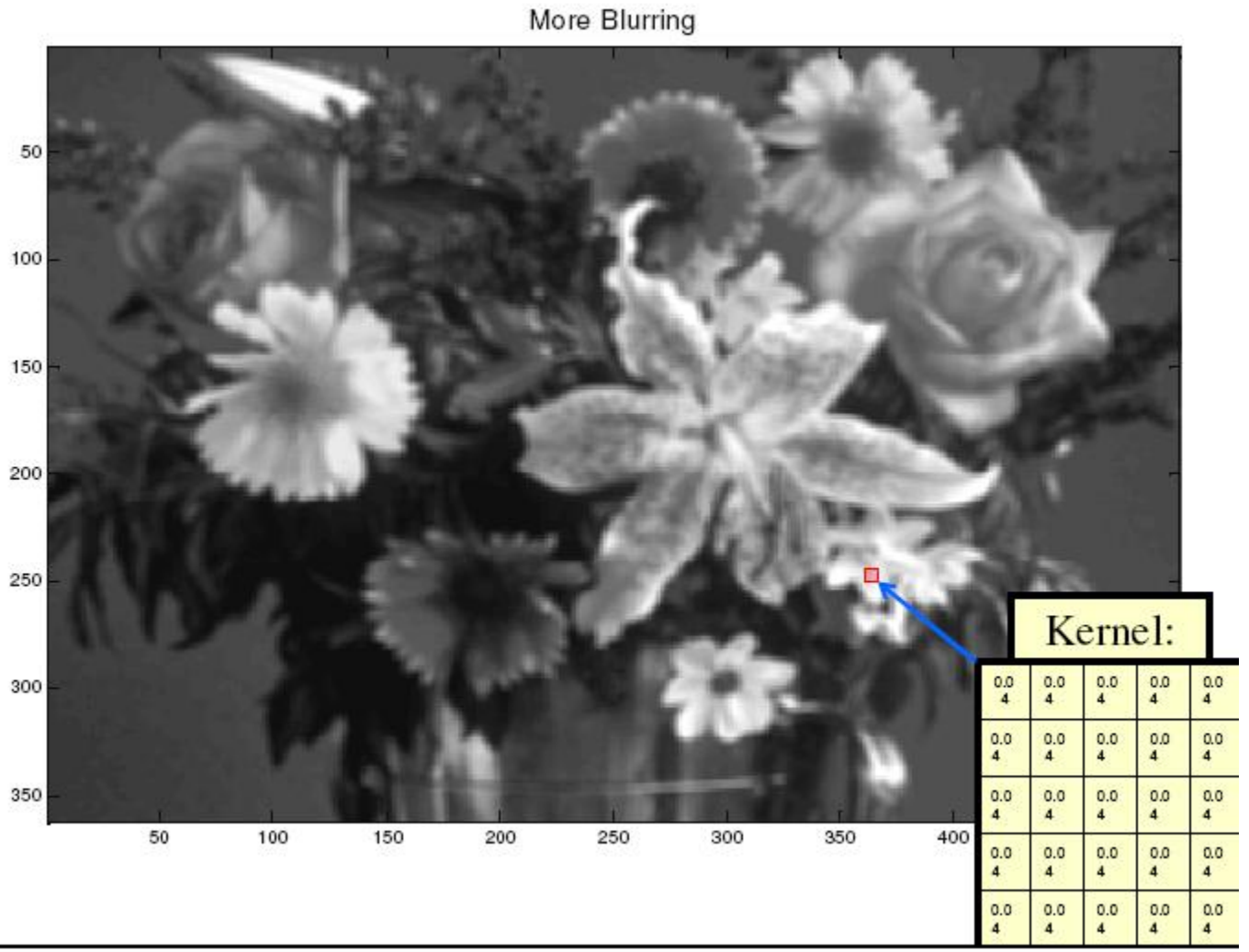
- Spatial filters
  - Use a **mask (kernel)** over an image region.
  - Work directly with pixels.
  - As opposed to: **Frequency filters**.
- Advantages
  - Simple implementation: **convolution** with the kernel function.
  - Different masks offer a **large variety of functionalities**.

Example (360x500 image)

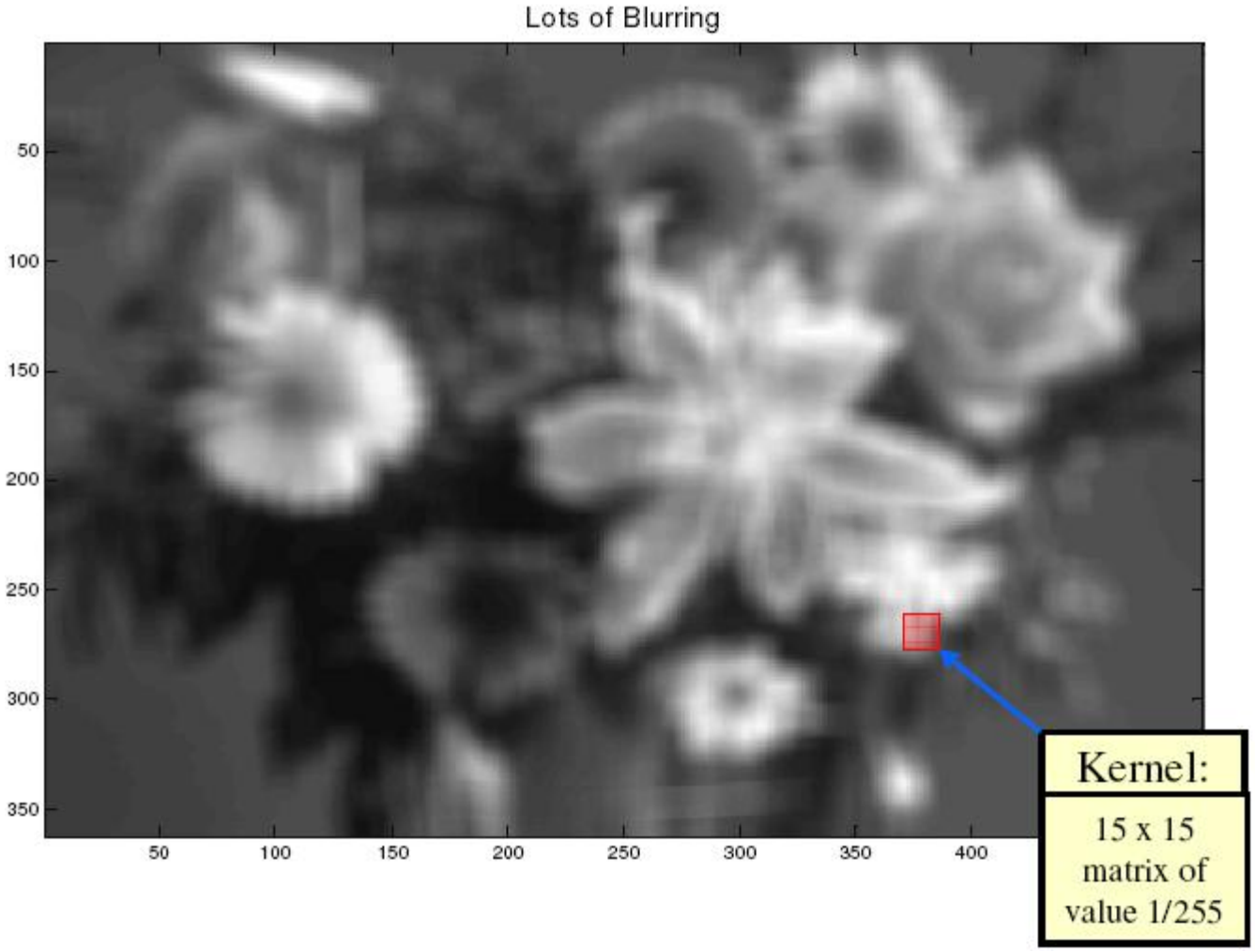
Example (360x500 image)



Example (360x500 image)



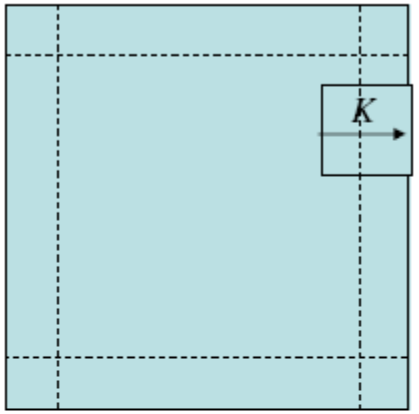
Example (360x500 image)



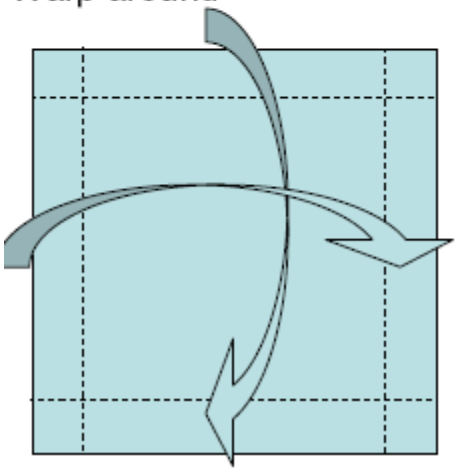
Convolution Border Issues

Practicalities (discrete convolution)

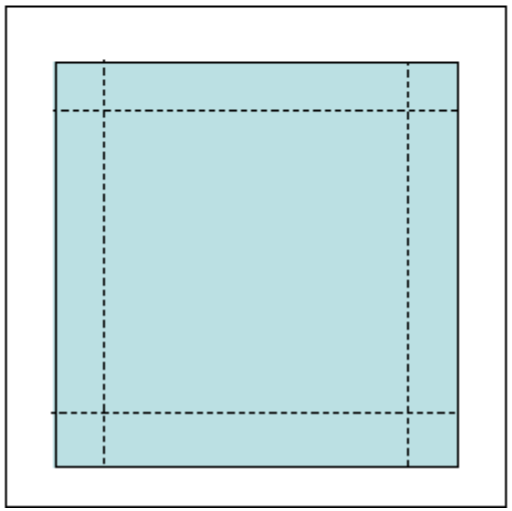
- MATLAB: `conv` (1D) or `conv2` (2D)
- Border issues:
  - When applying convolution with a KxK kernel, the result is undefined for pixels closer than K pixels from the border of the image
- Options:



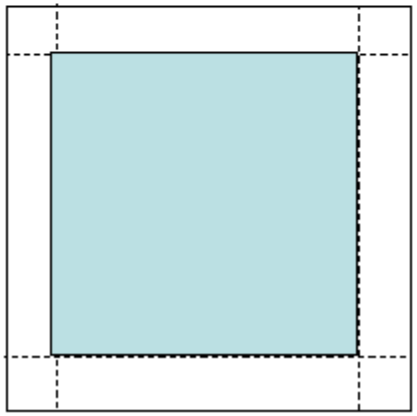
Warp around



Expand/Pad



Crop



Gaussian Smoothing

- A Gaussian kernel gives less weight to pixels further from the center of the window

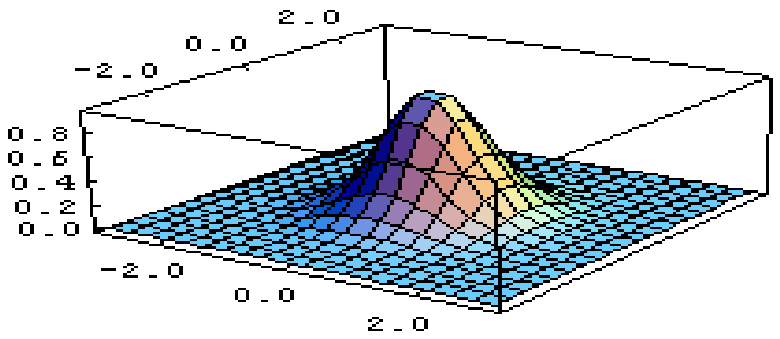
$$H[u, v]$$

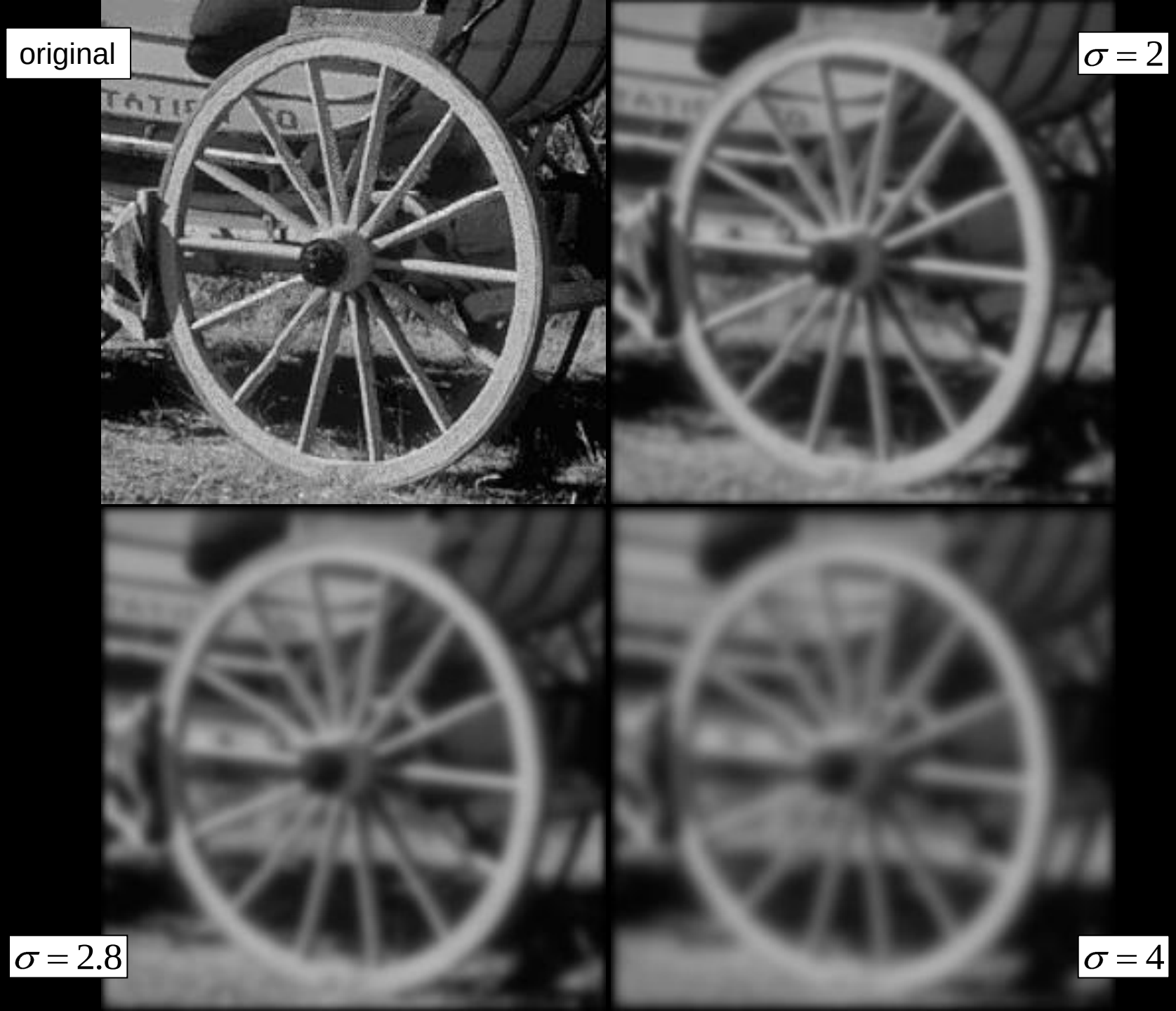
$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

- This kernel is an approximation of a Gaussian function:

$$F[x, y]$$

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

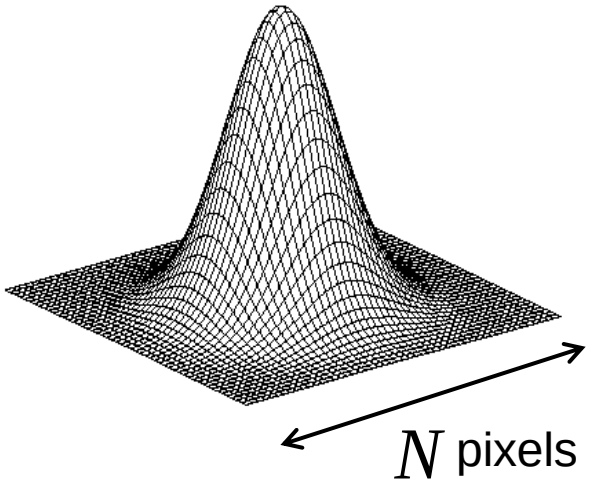



original
$\sigma = 2$
$\sigma = 2.8$
$\sigma = 4$

Gaussian Smoothing (maths)

Gaussian kernel

$$h(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$



Filter size $N \propto \sigma$ …can be very large (truncate, if necessary)

$$g(i,j) = \frac{1}{2\pi\sigma^2} \sum_{m=1} \sum_{n=1} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} f(i-m, j-n)$$

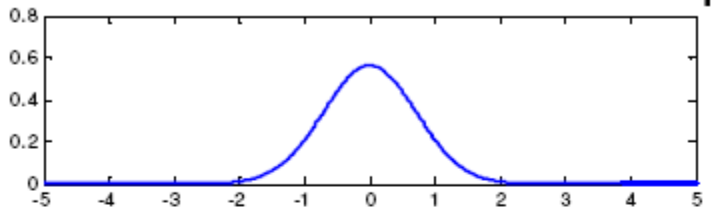
2D Gaussian is separable!

$$g(i,j) = \frac{1}{2\pi\sigma^2} \sum_{m=1} e^{-\frac{1}{2}\frac{m^2}{\sigma^2}} \sum_{n=1} e^{-\frac{1}{2}\frac{n^2}{\sigma^2}} f(i-m, j-n)$$

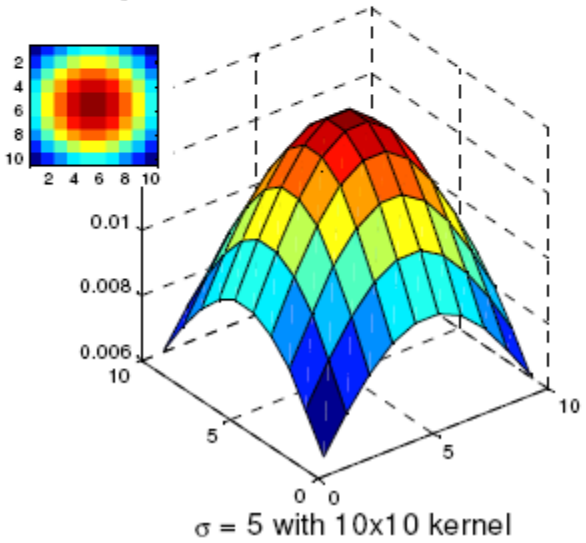
Use two 1D Gaussian Filters!

Finite kernel support

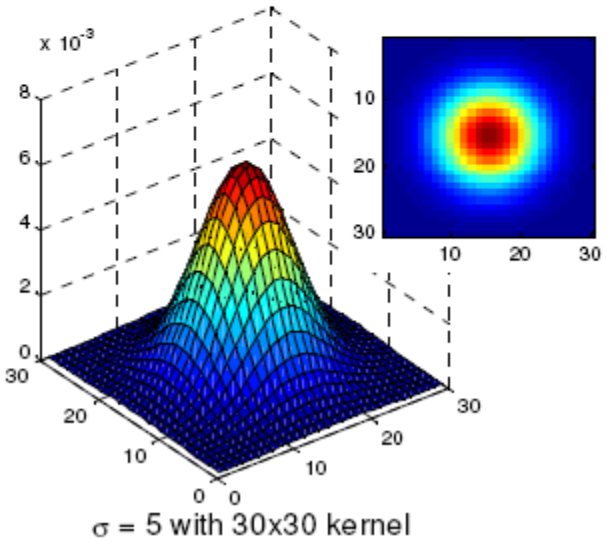
- Gaussian function has infinite support



- In discrete filtering, we have finite kernel



σ = 5 with 10x10 kernel

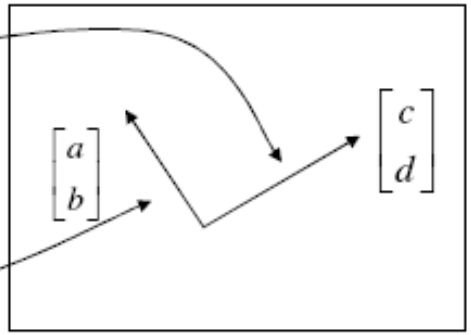


σ = 5 with 30x30 kernel

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 $[c\ d]$

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

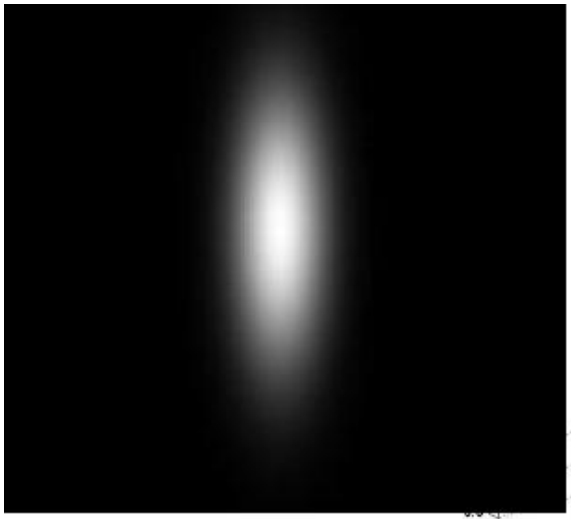


- We can write this in a more compact form by using the standard multivariate Gaussian notation:

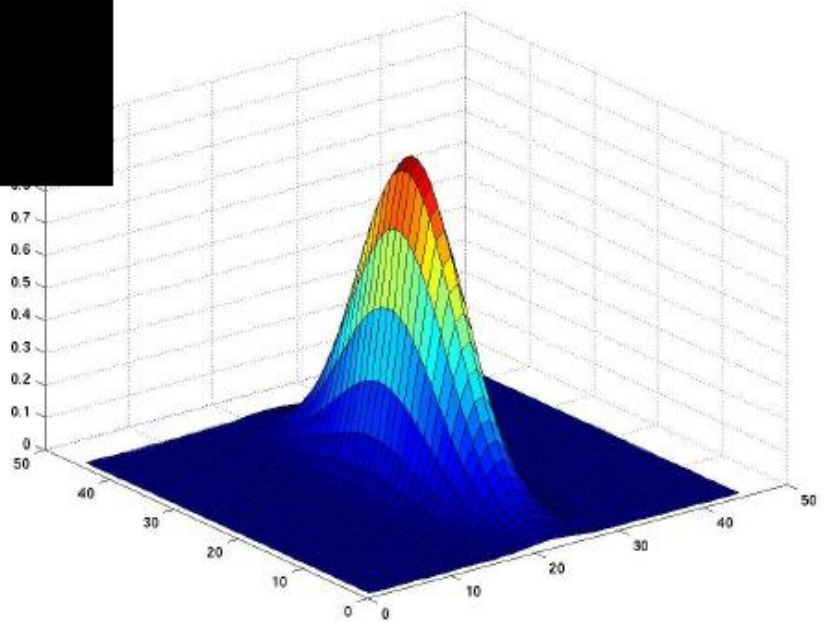
$$G_\Sigma = e^{-\frac{X^T\Sigma^{-1}X}{2}} \qquad X = \begin{bmatrix} x \\ y \end{bmatrix}$$

- 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.

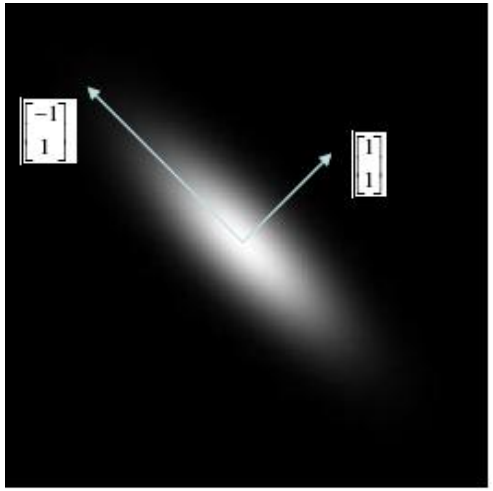
Oriented Gaussian Filters



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

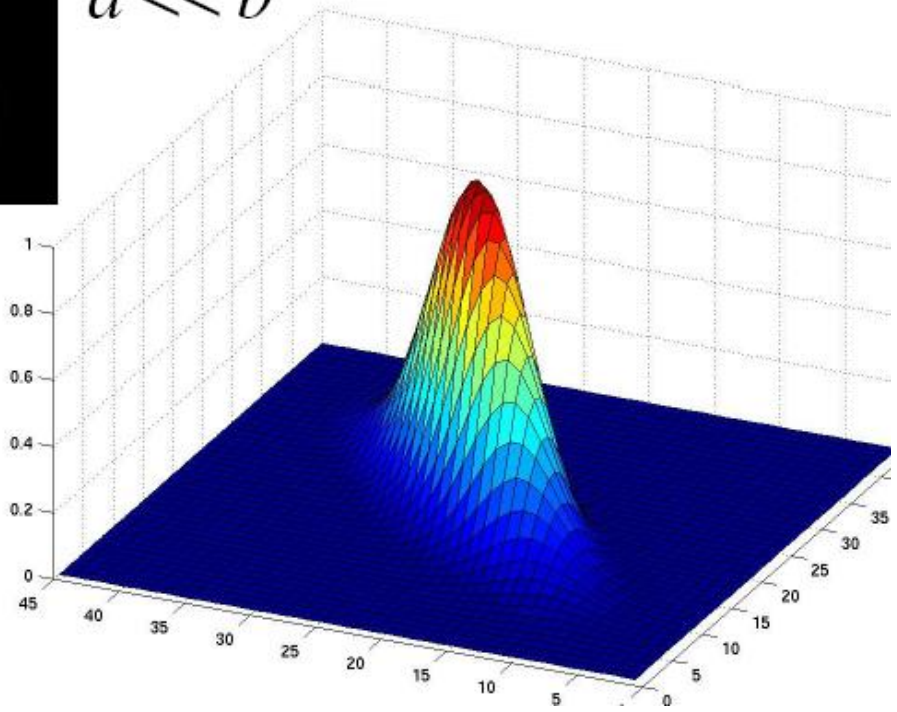


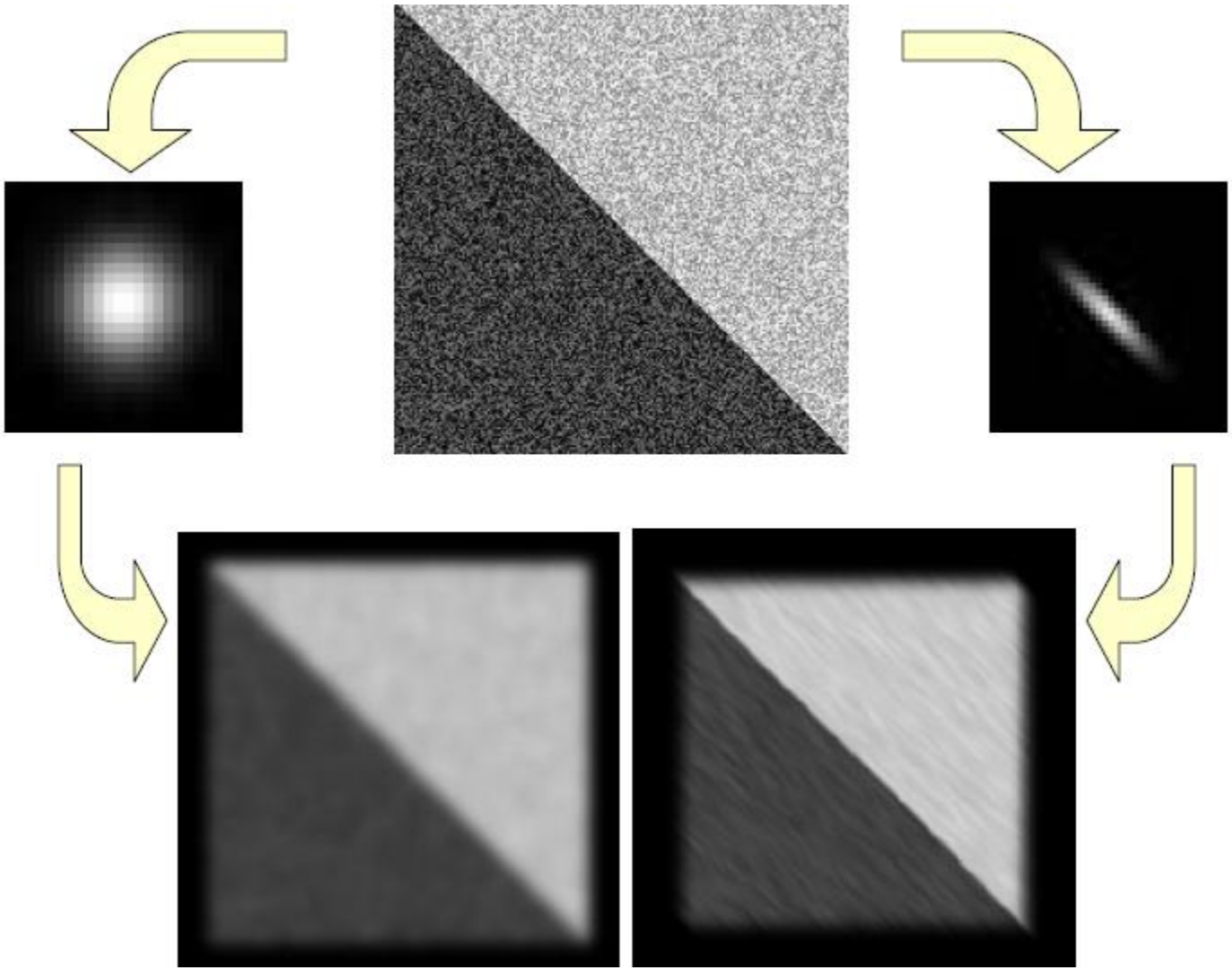
Oriented Gaussian Filters



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

$$a << b$$



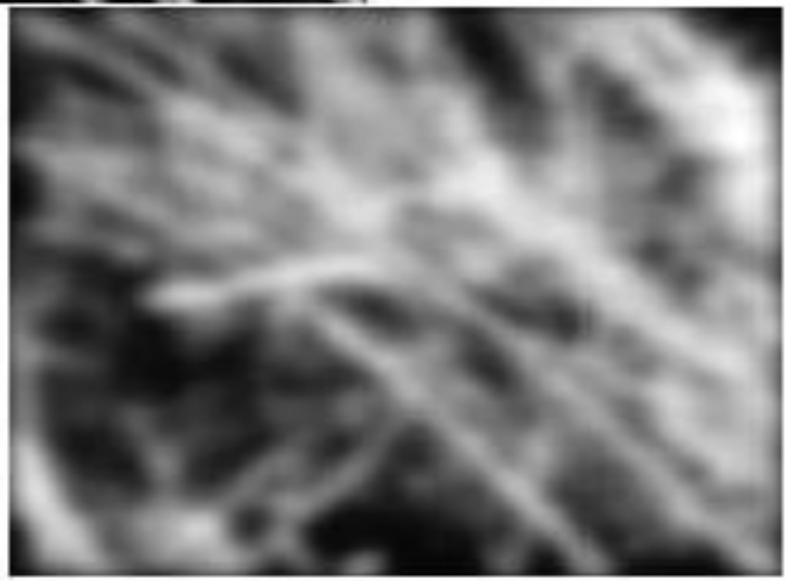


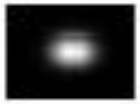
Mean Filtering

- We are degrading the energy of the high spatial frequencies of an image (**low-pass filtering**).
  - Makes the image ‘smoother’.
  - Used in noise reduction.
- Can be implemented with spatial masks or in the frequency domain.

1	1	1
1	1	1
1	1	1

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





Mean filter

Gaussian filter

by Charles Allen Gillbert

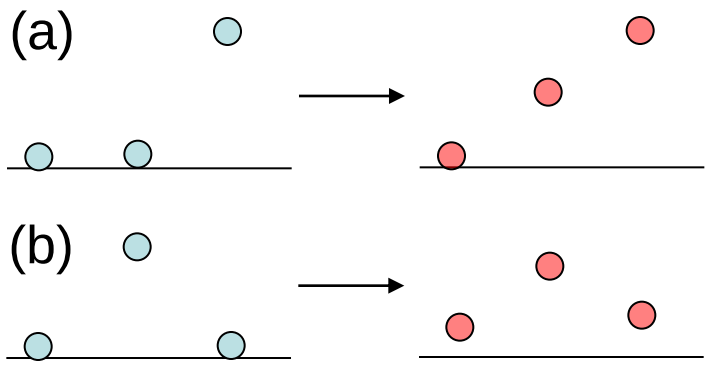
by Harmon & Julesz

http://www.michaelbach.de/ot/cog_blureffects/index.html

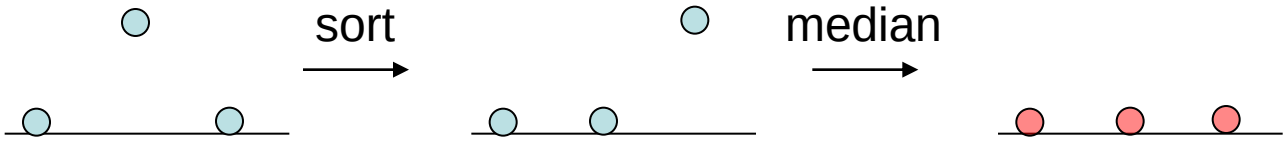
http://www.michaelbach.de/ot/cog_blureffects/index.html

Median Filter

- Smoothing is averaging
  - (a) Blurs edges
  - (b) Sensitive to outliers



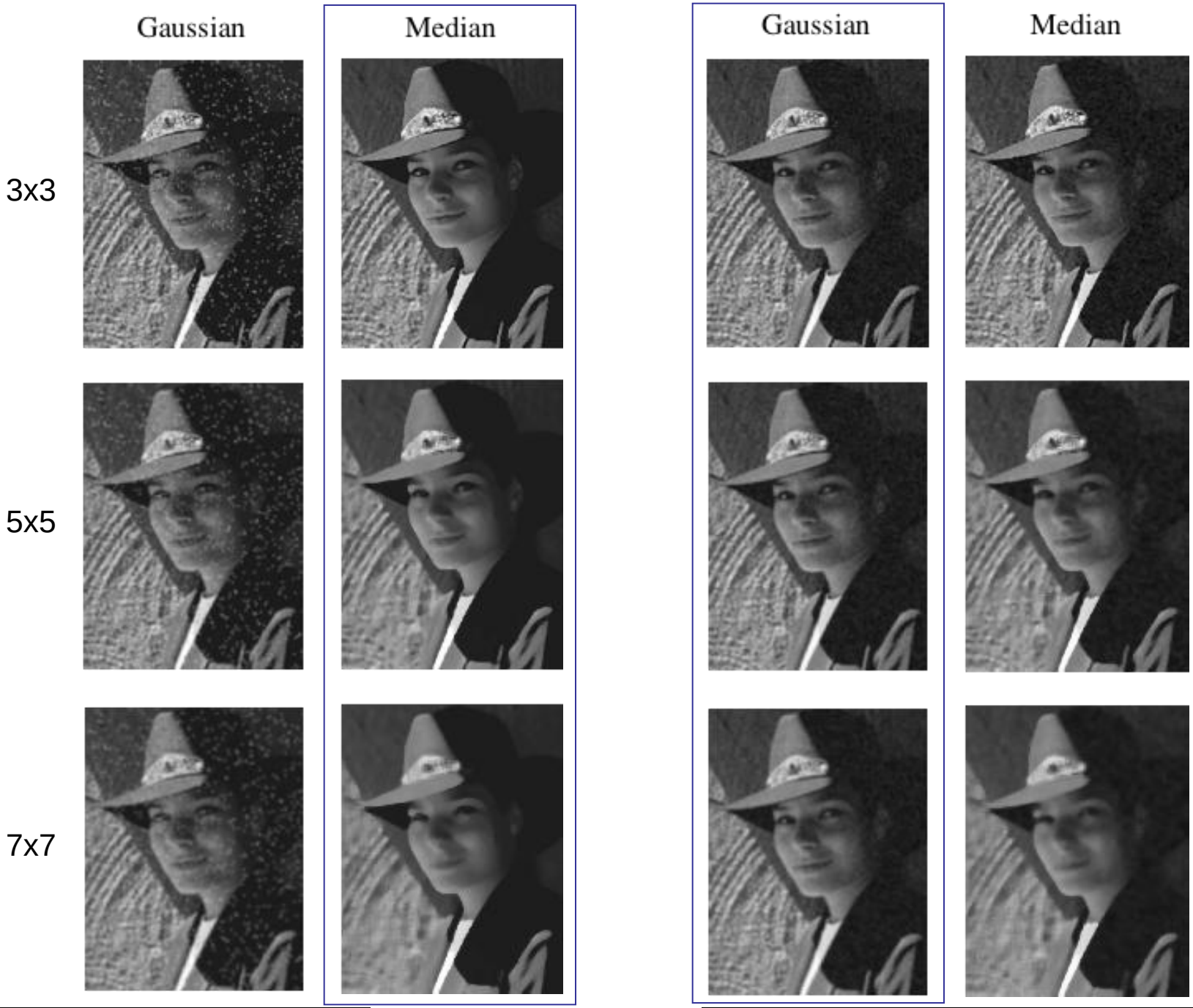
- Median filtering
  - Sort N^2-1 values around the pixel
  - Select middle value (median)



  - Non-linear (Cannot be implemented with convolution)

Salt and pepper noise

Gaussian noise



Topic: Frequency domain filtering

- Frequency Space
- Spatial Convolution
- Spatial filters
- Frequency domain filtering
- Edge detection

Image Processing in the Fourier Domain

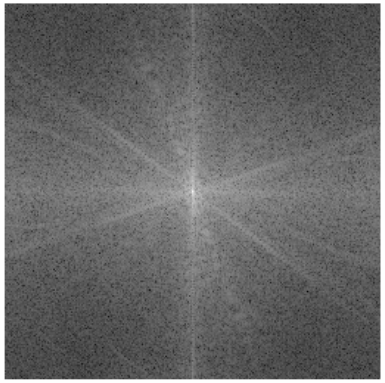
Magnitude of the FT

Does not look anything like what we have seen

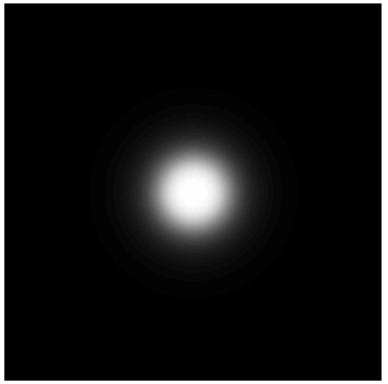
Low-pass Filtering

Original image

FFT of original image

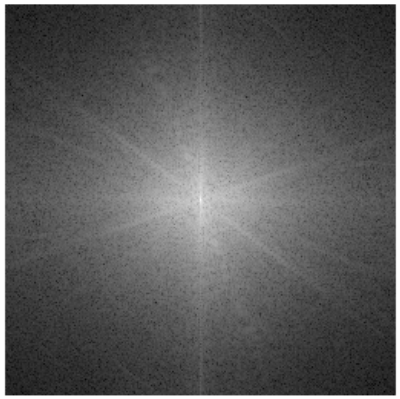


Low-pass filter



Low-pass image

FFT of low-pass image



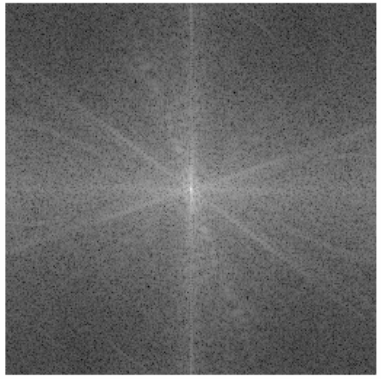
Lets the low frequencies pass and eliminates the high frequencies.

Generates image with overall shading, but not much detail

High-pass Filtering

Original image

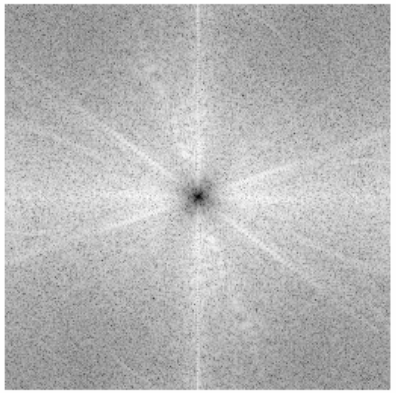
FFT of original image



High–pass filter

High–pass image

FFT of high–pass image

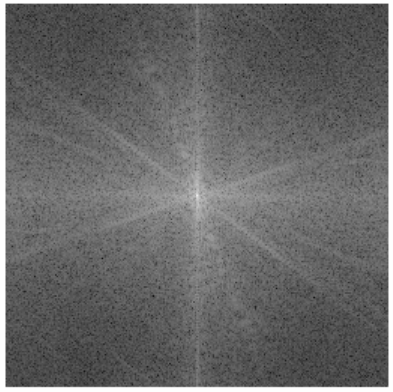


Lets through the high frequencies (the detail), but eliminates the low frequencies (the overall shape). It acts like an edge enhancer.

Boosting High Frequencies

Original image

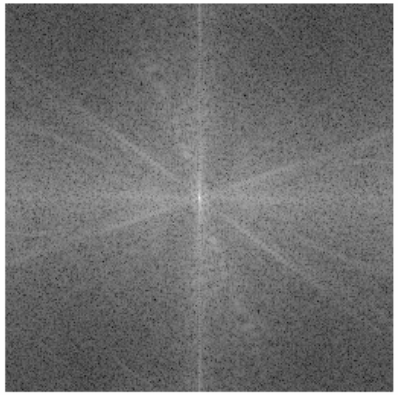
FFT of original image



High–boost filter

High boosted image

FFT of high boosted image

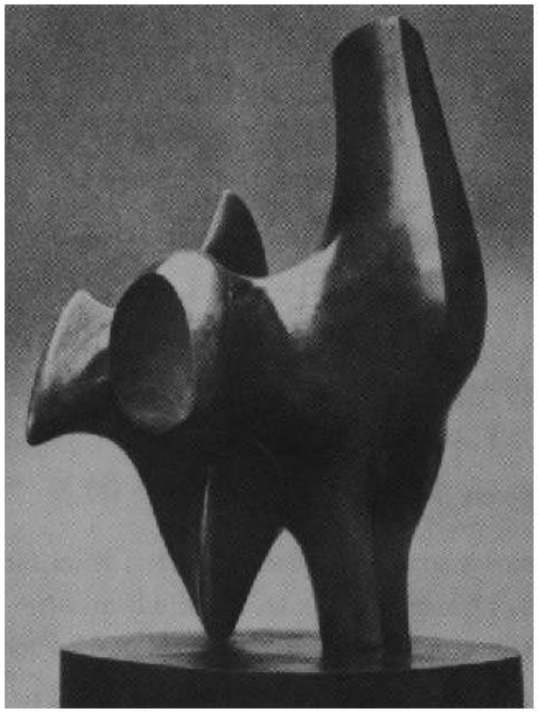


Topic: Edge detection

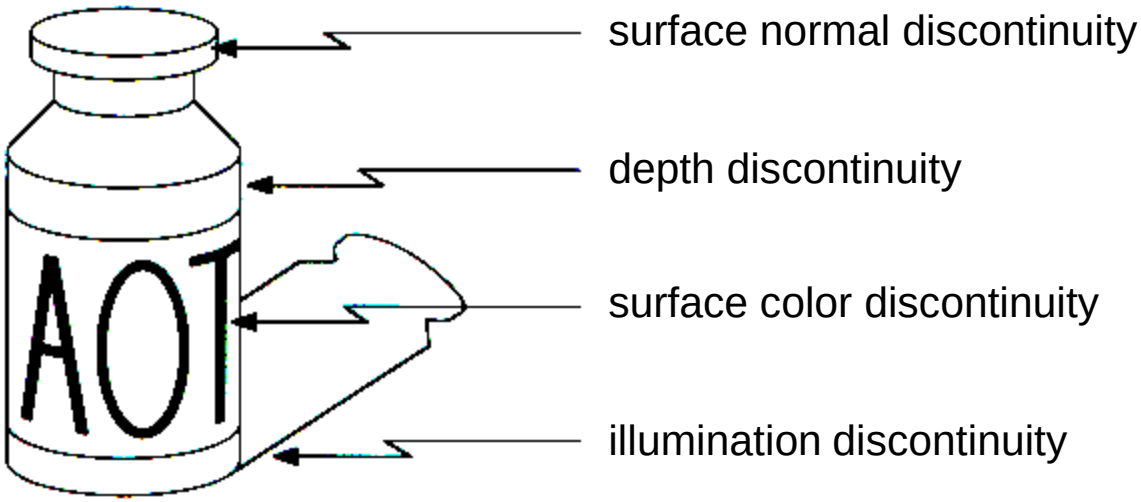
- Frequency Space
- Spatial Convolution
- Spatial filters
- Frequency domain filtering
- Oriented Gaussian filters
- Edge detection

Edge Detection

- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

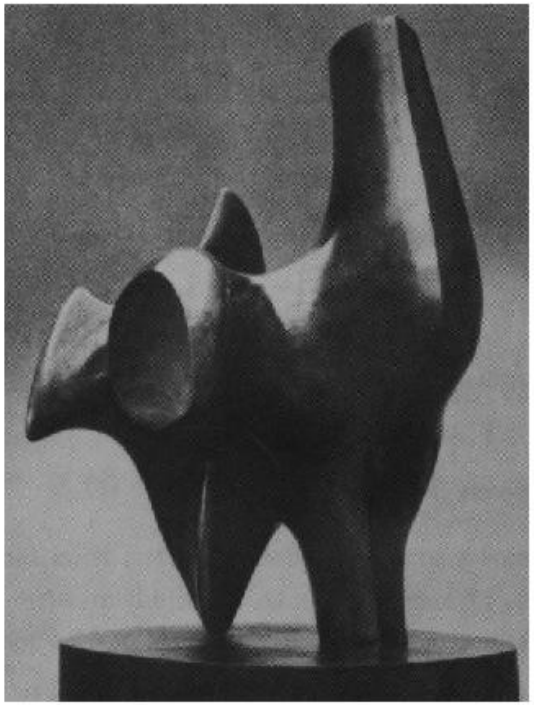


Origin of Edges



- Edges are caused by a variety of factors

How can you tell that a pixel is on an edge?



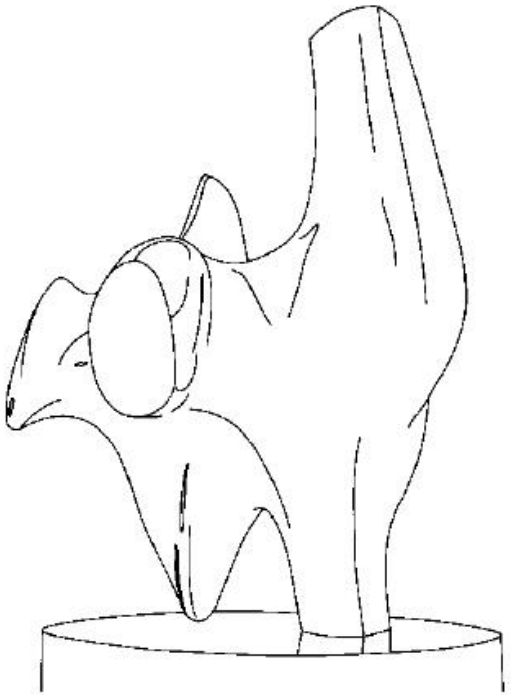


Image derivatives

- 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:

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

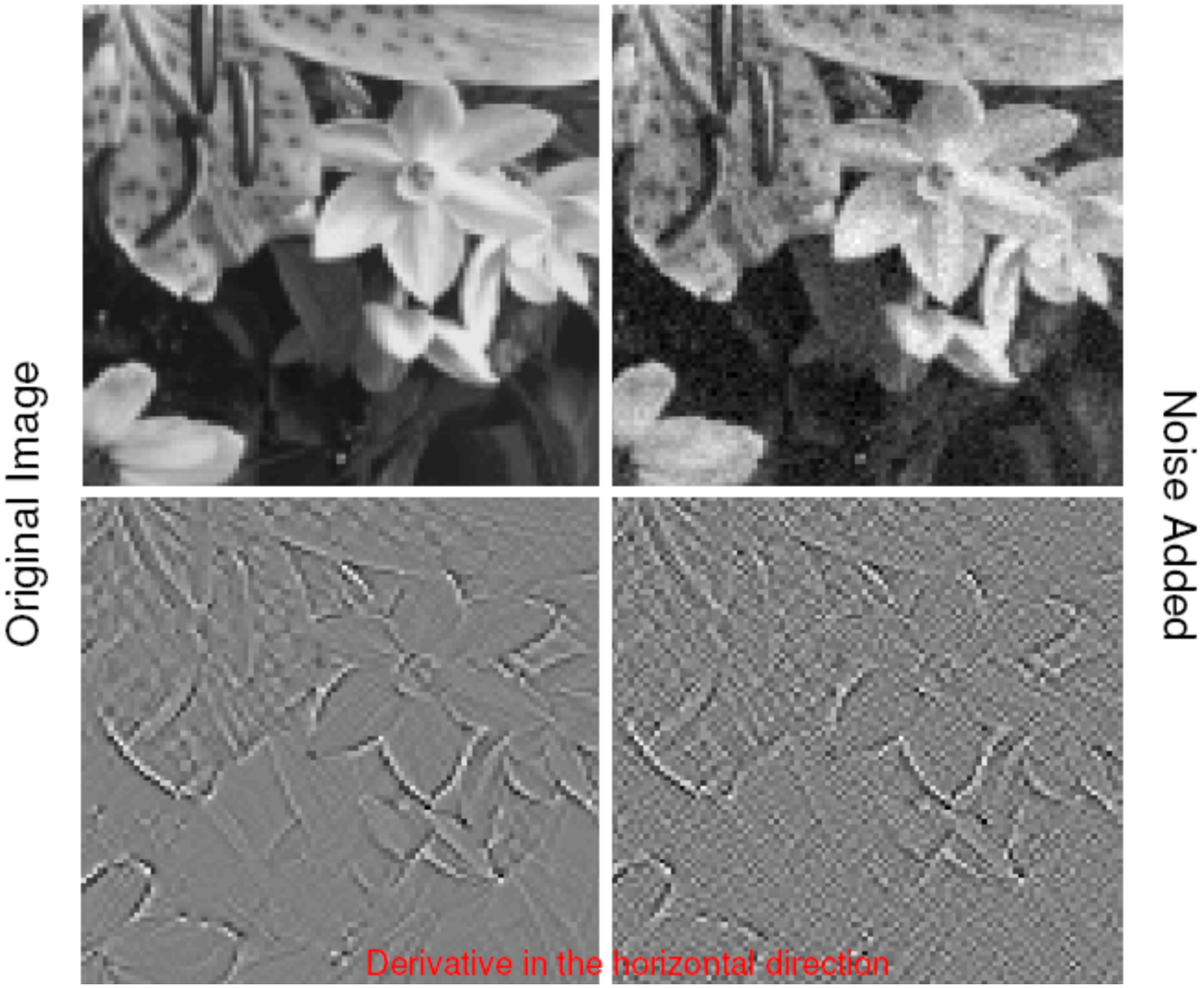
- Convolution of the image by

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

- Problem: Increases noise

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

Image derivatives



Smooth derivative

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

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

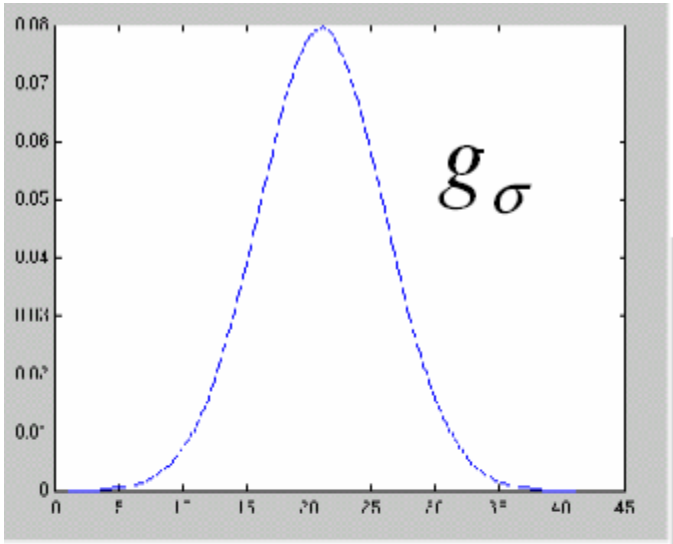
- Applying the differentiation property of the convolution:

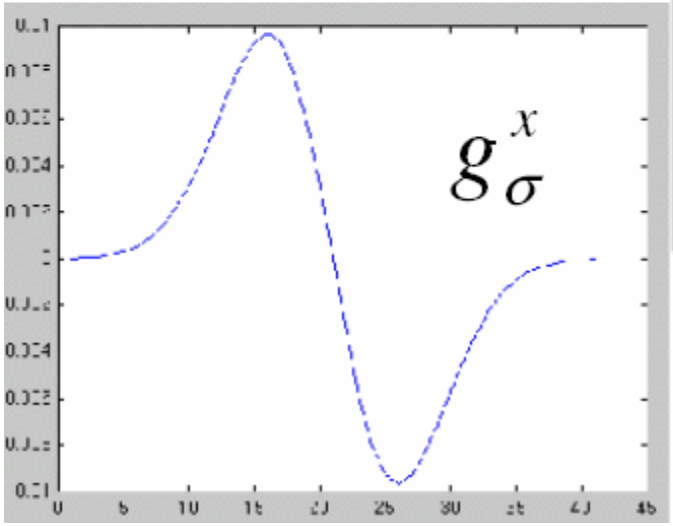
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

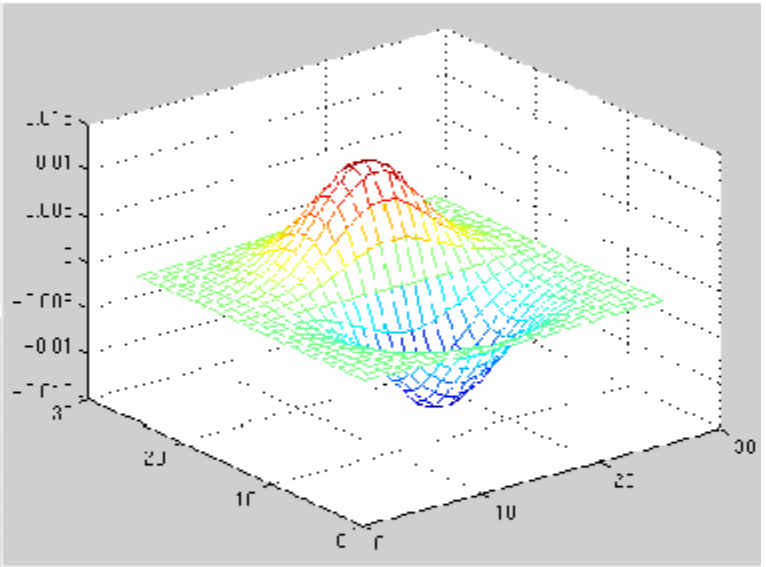
- Therefore, taking the derivative in x of the image can be done by convolution with the derivative of a Gaussian:

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

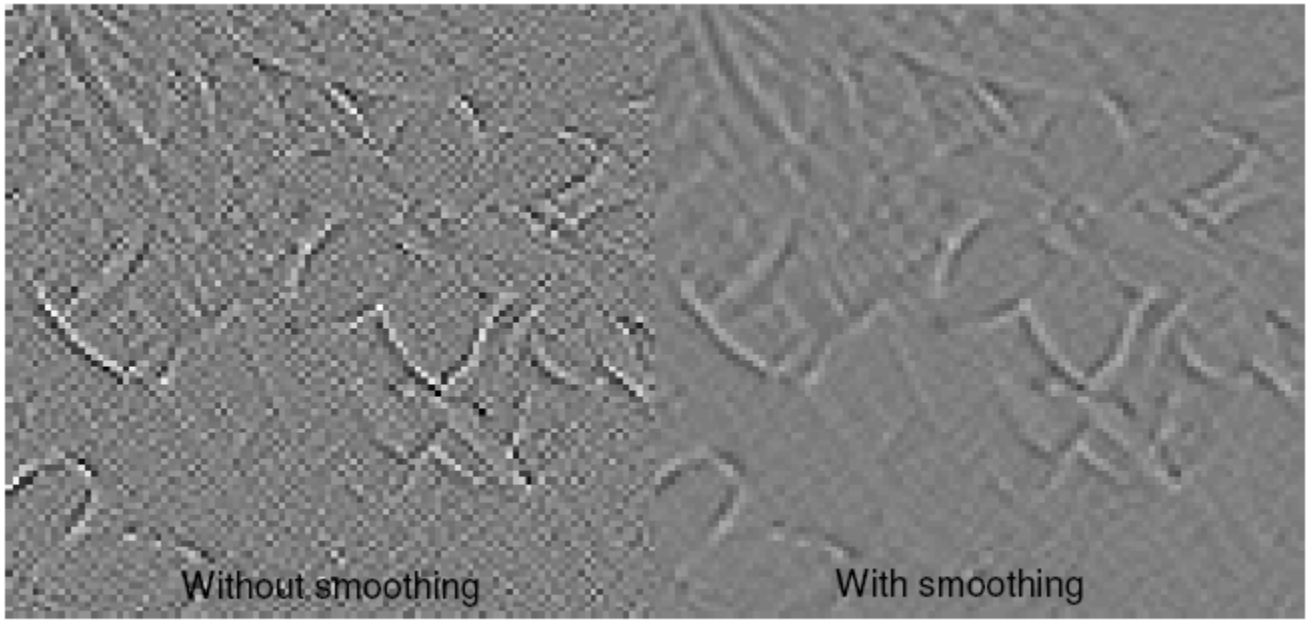
Smooth derivative







Smooth derivative

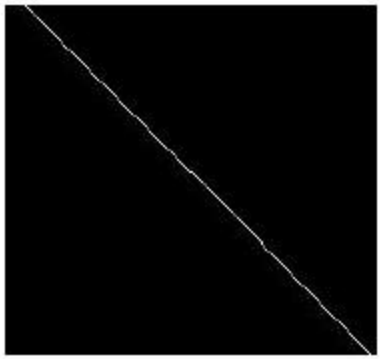


Better but still blurs away edge information

Smooth but …

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

Image with Edge



Edge Location

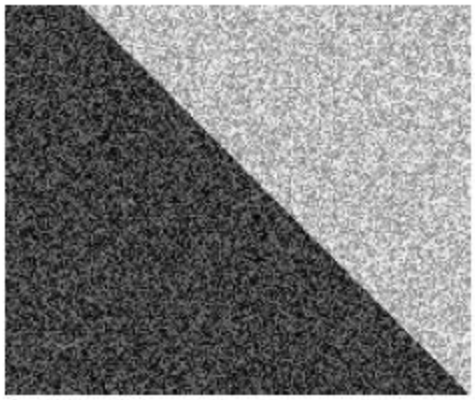
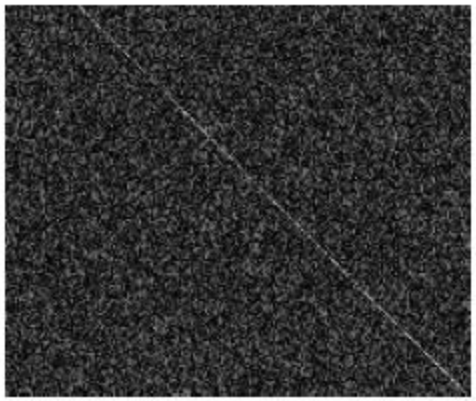
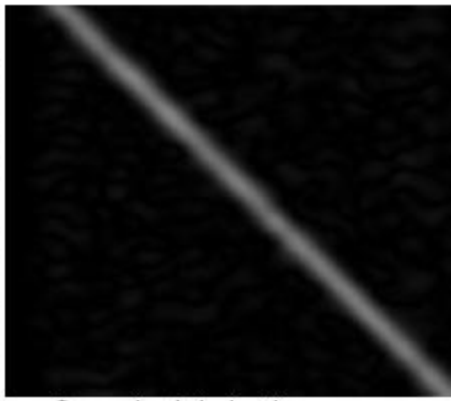


Image + Noise



Derivatives detect edge *and* noise



Smoothed derivative removes noise, but blurs edge

Gradient

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

- The *edge strength* is given by the gradient magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

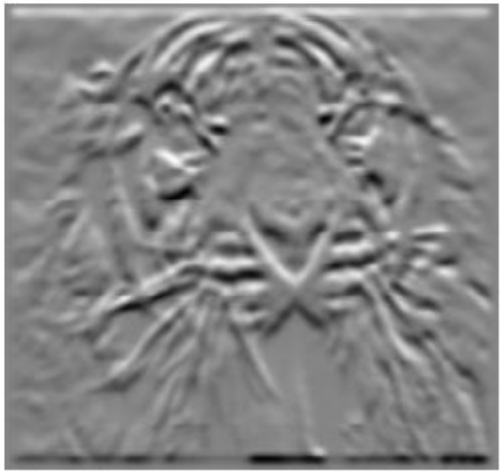
Gradient

Applying the first derivative of Gaussian

I

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

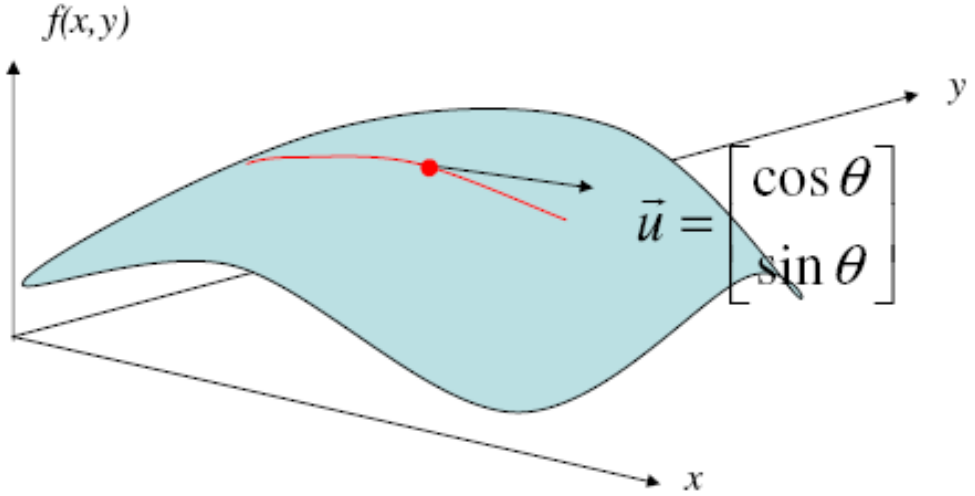
$\frac{\partial I}{\partial x}$



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

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:



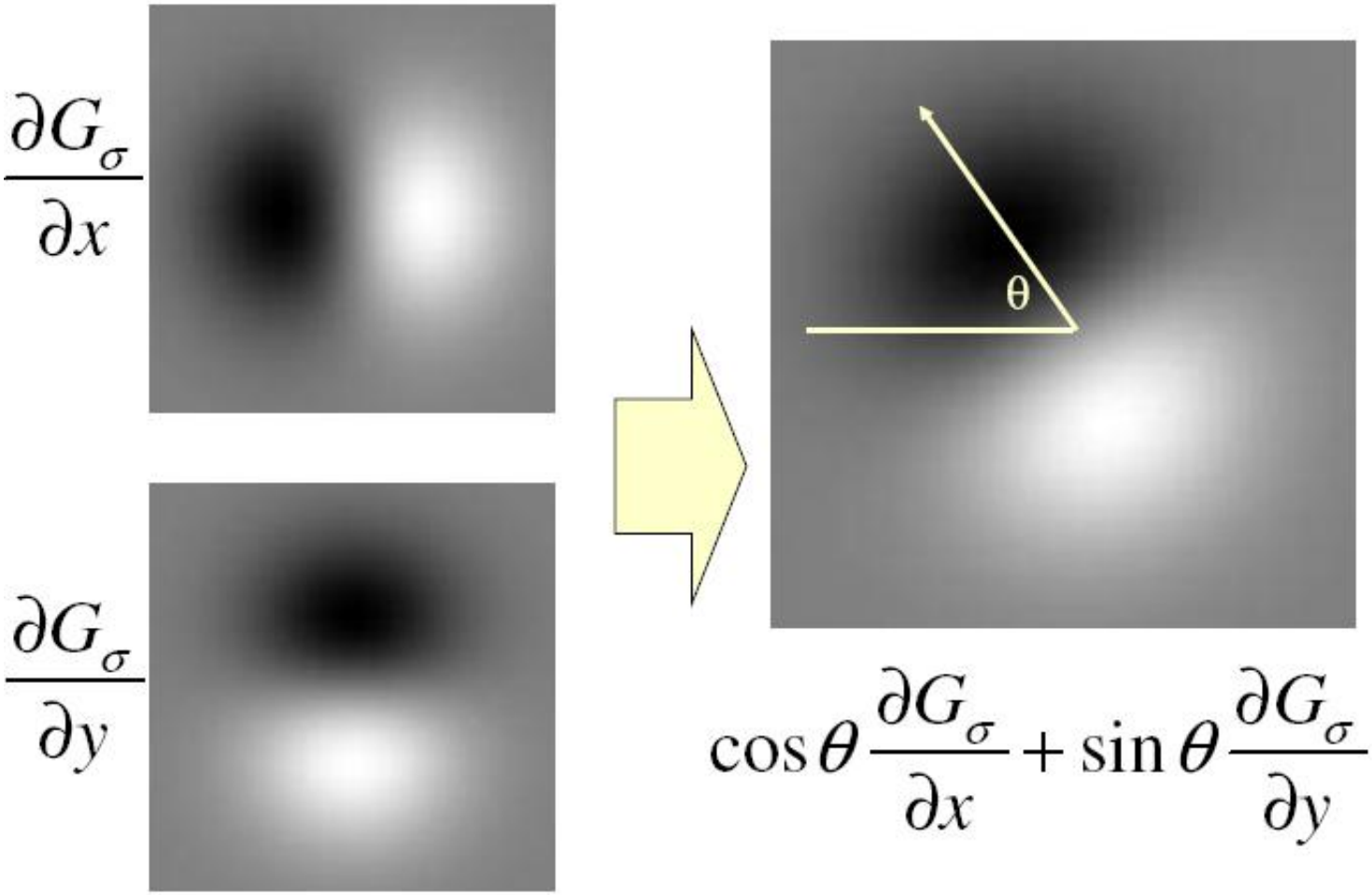
- Conveniently, the derivative is obtained by convolution of the image with a simple linear combination of the axis derivatives:

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

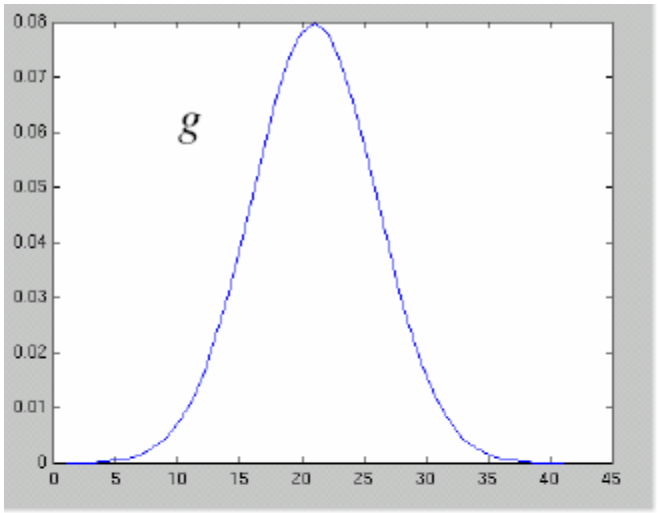
- Note: More generally, filters that can be computed at any orientation as a linear combination of other, fixed filters are called steerable filters

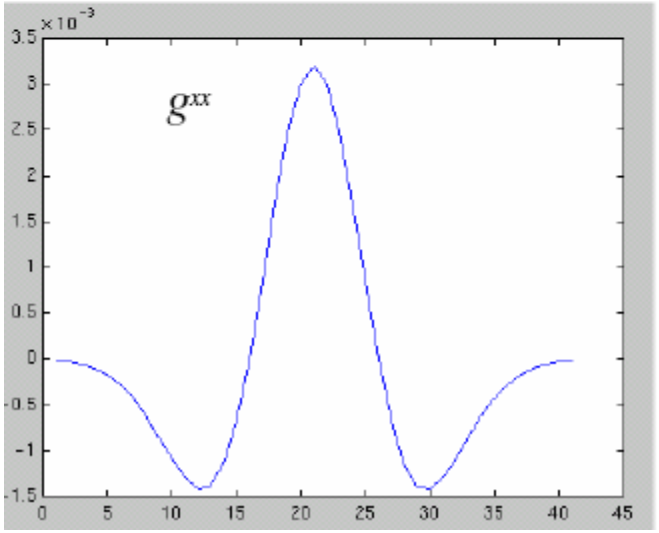
$$F(\theta) = \sum a_i(\theta) F_i$$

Directional derivatives

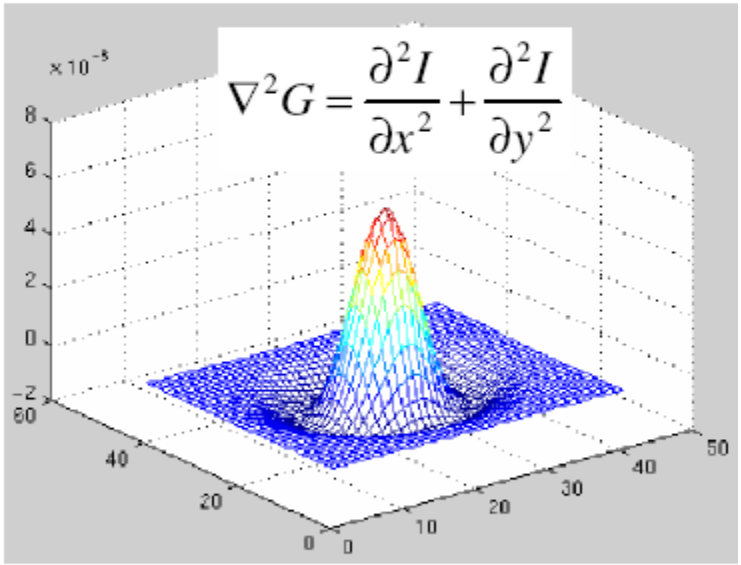


Laplacian Operator





Second derivatives:
Laplacian

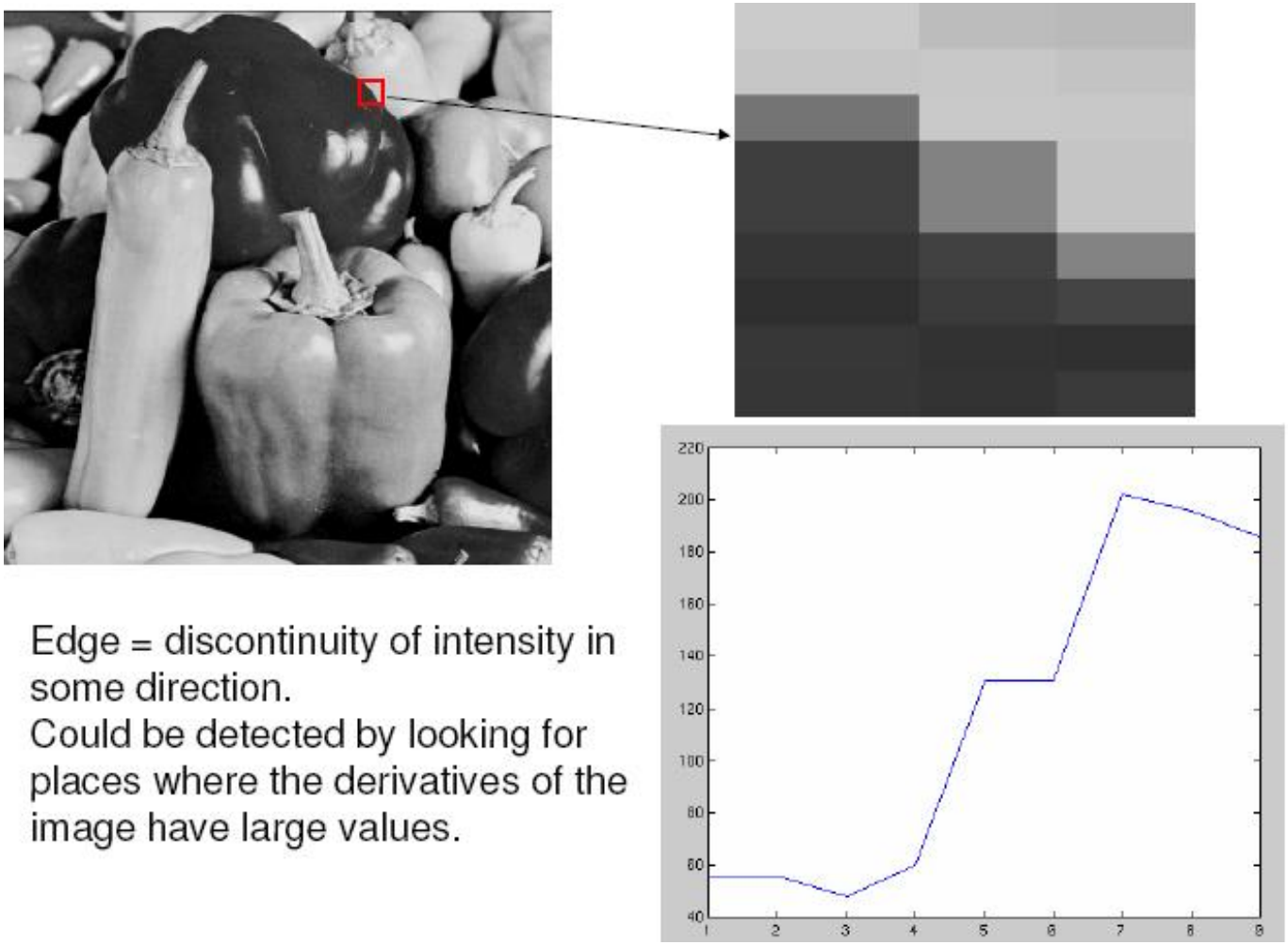


Operators for edge detection

Edge Detection

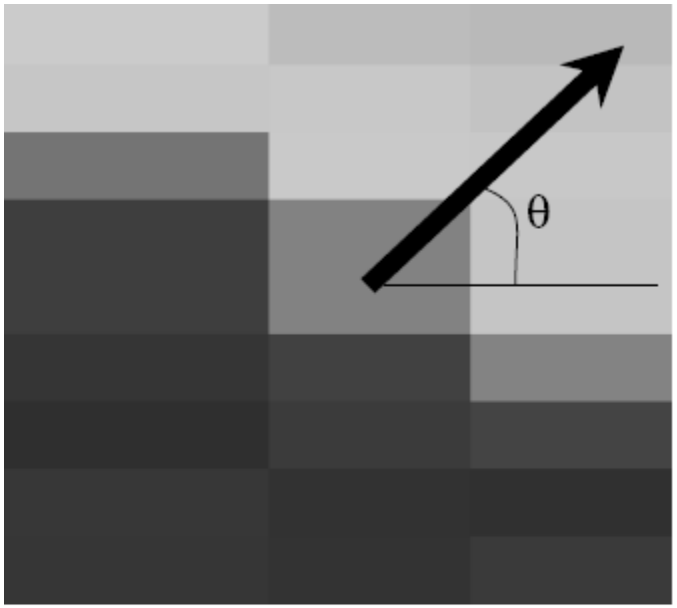
- Edge Detection
  - Gradient operators
  - Canny edge detectors
  - Laplacian detectors

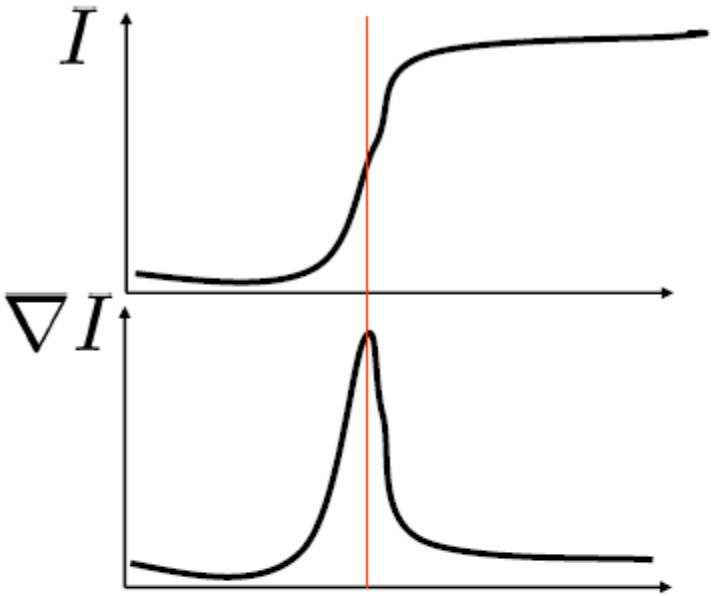
What is an edge ?



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

Gradient of an edge



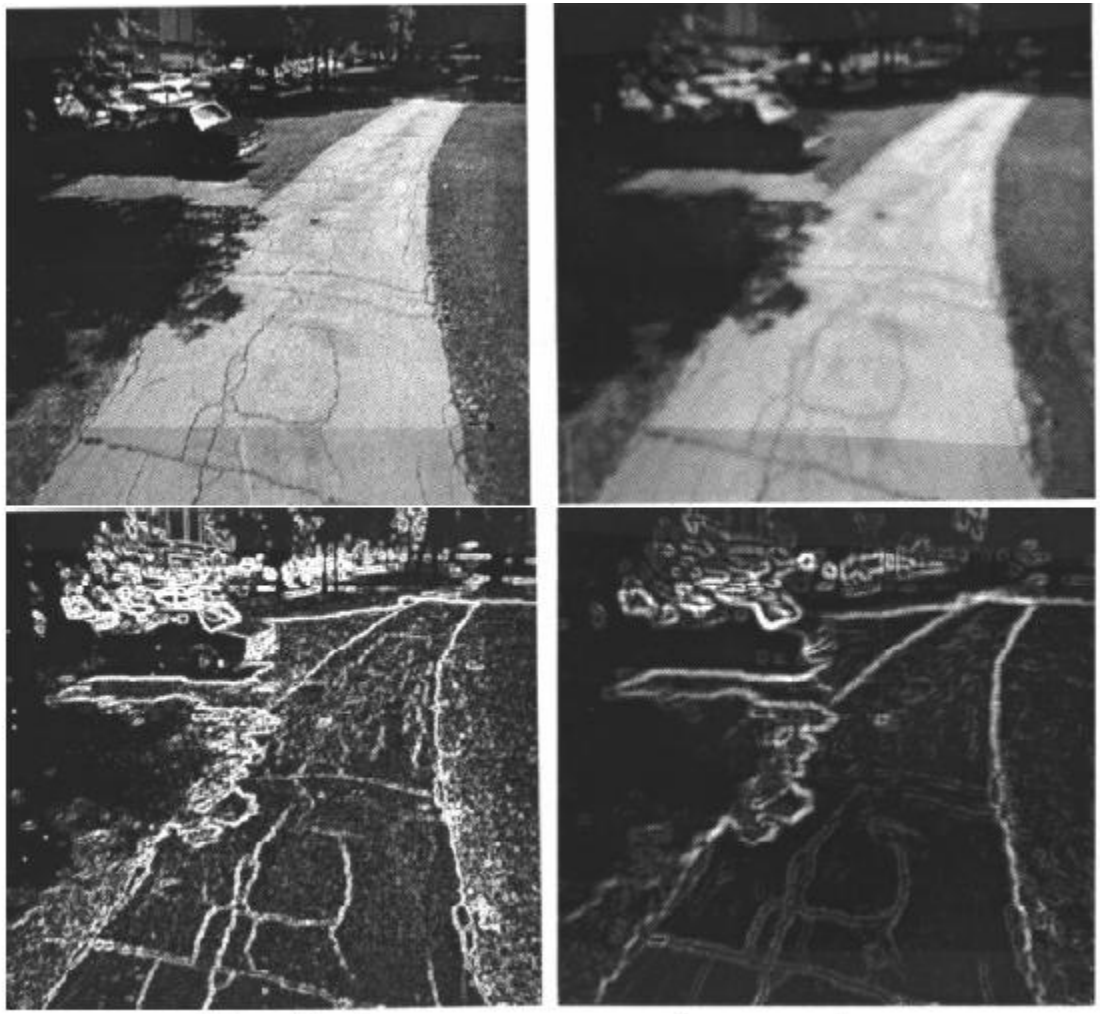


Edge pixels are at local maxima of gradient magnitude
Gradient computed by convolution with Gaussian derivatives
Gradient direction is always perpendicular to edge direction

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

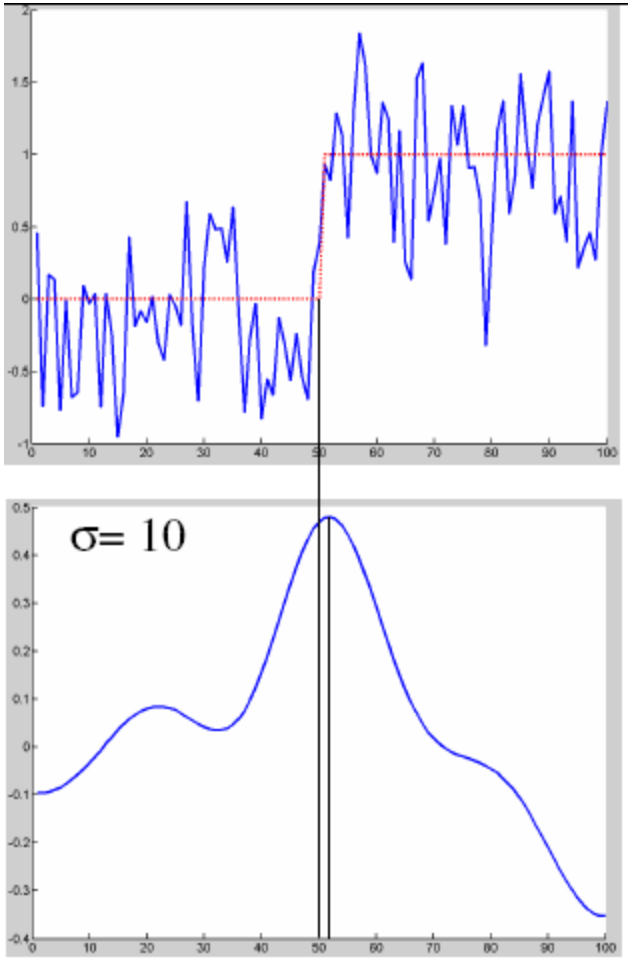
$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \qquad \theta = atan2(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x})$$

Gradient operator for edge detection

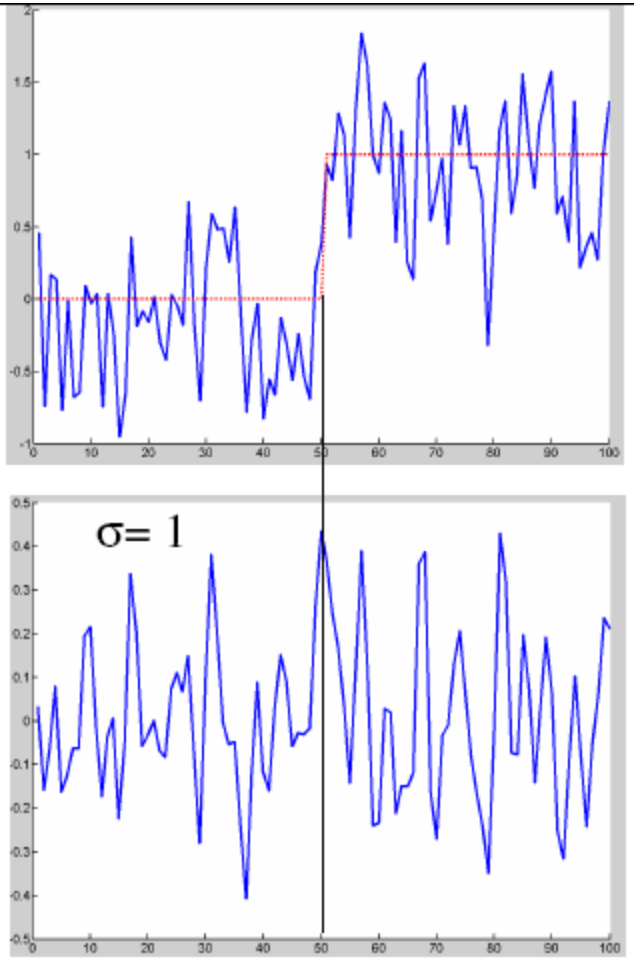


Small sigma Large sigma

Gradient operator for edge detection



Large σ → Good detection (high SNR)
Poor localization



Small σ → Poor detection (low SNR)
Good localization

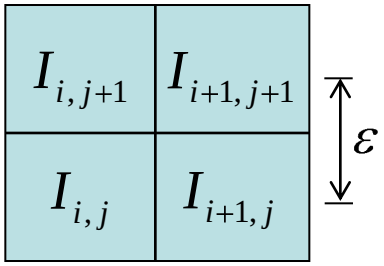
Discrete Edge Operators

- How can we differentiate a ***discrete*** image?

  Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$



  Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Discrete Edge Operators

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- **Laplacian** :

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

Convolution masks :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array}$$

or

$$\frac{1}{6\varepsilon^2}\begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array}$$

(more accurate)

The Sobel Operators

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

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

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

Comparing Edge Operators

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

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

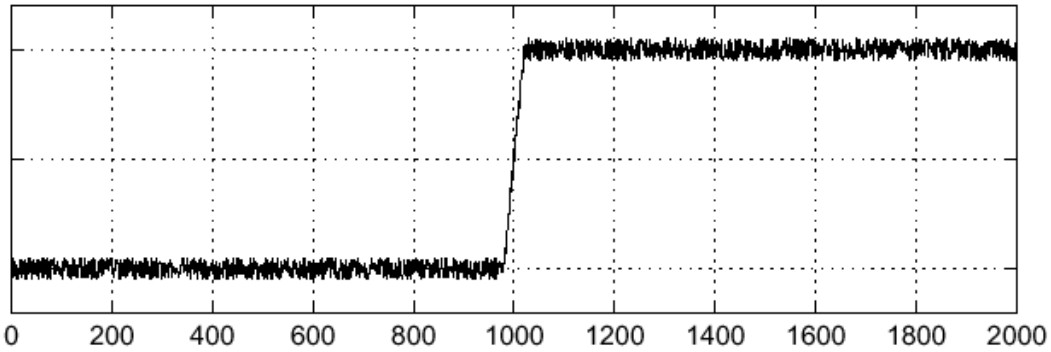
1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

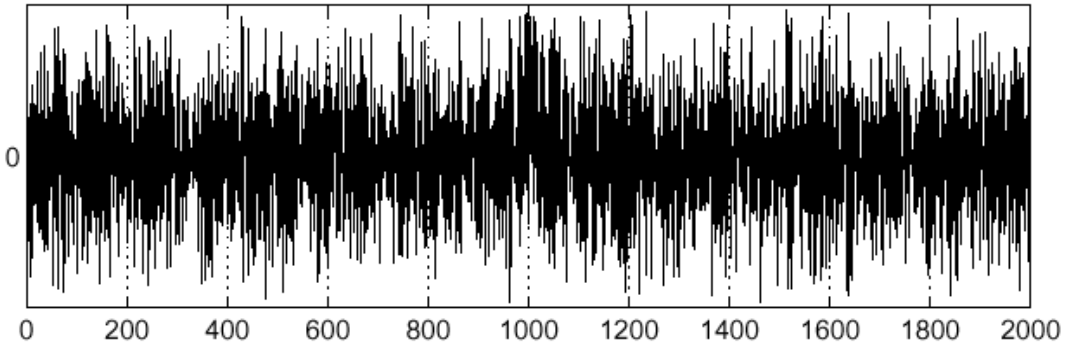
Effects of Noise

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

$f(x)$

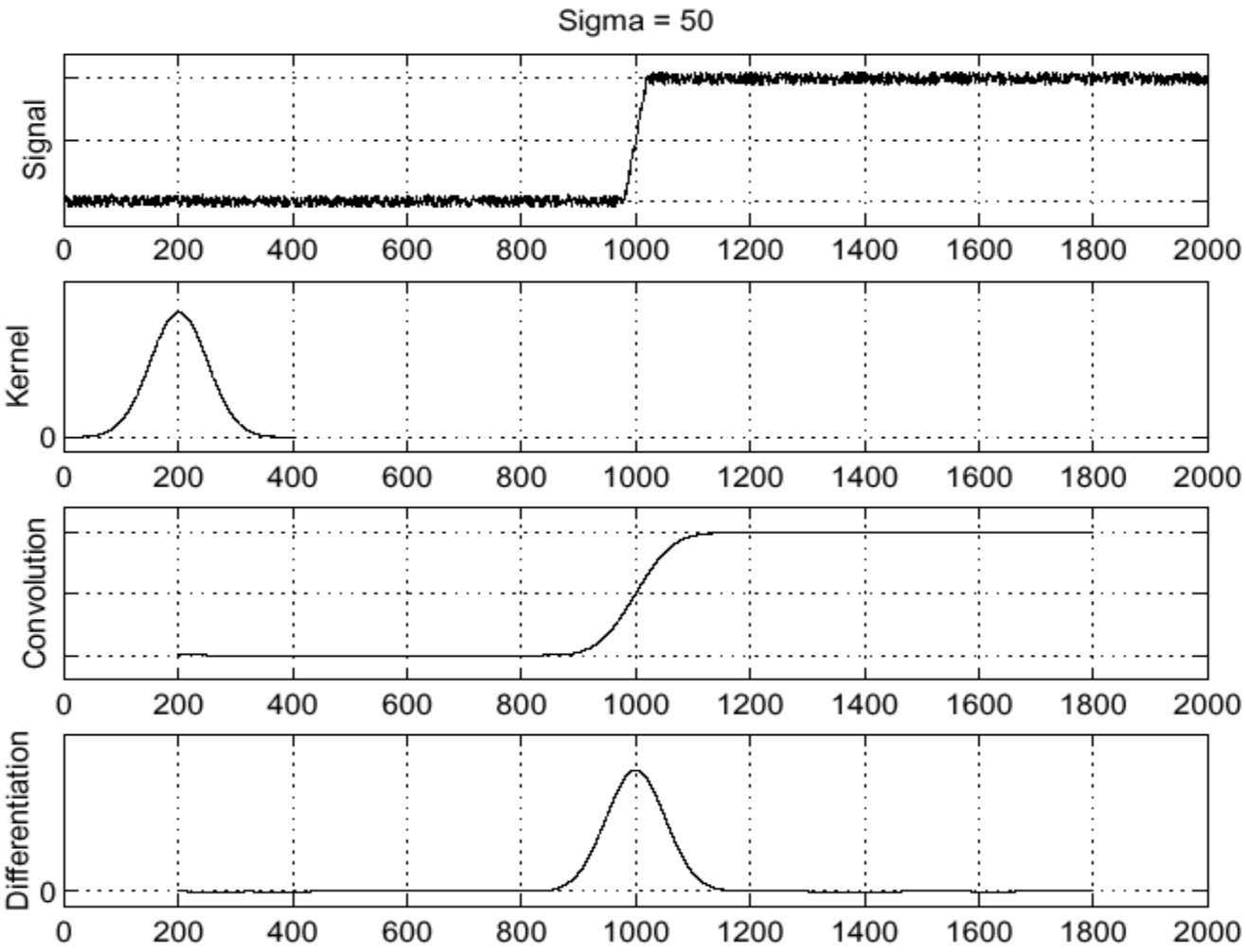


$\frac{d}{dx} f(x)$



Where is the edge??

Solution: Smooth First



$h \star f$

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

Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

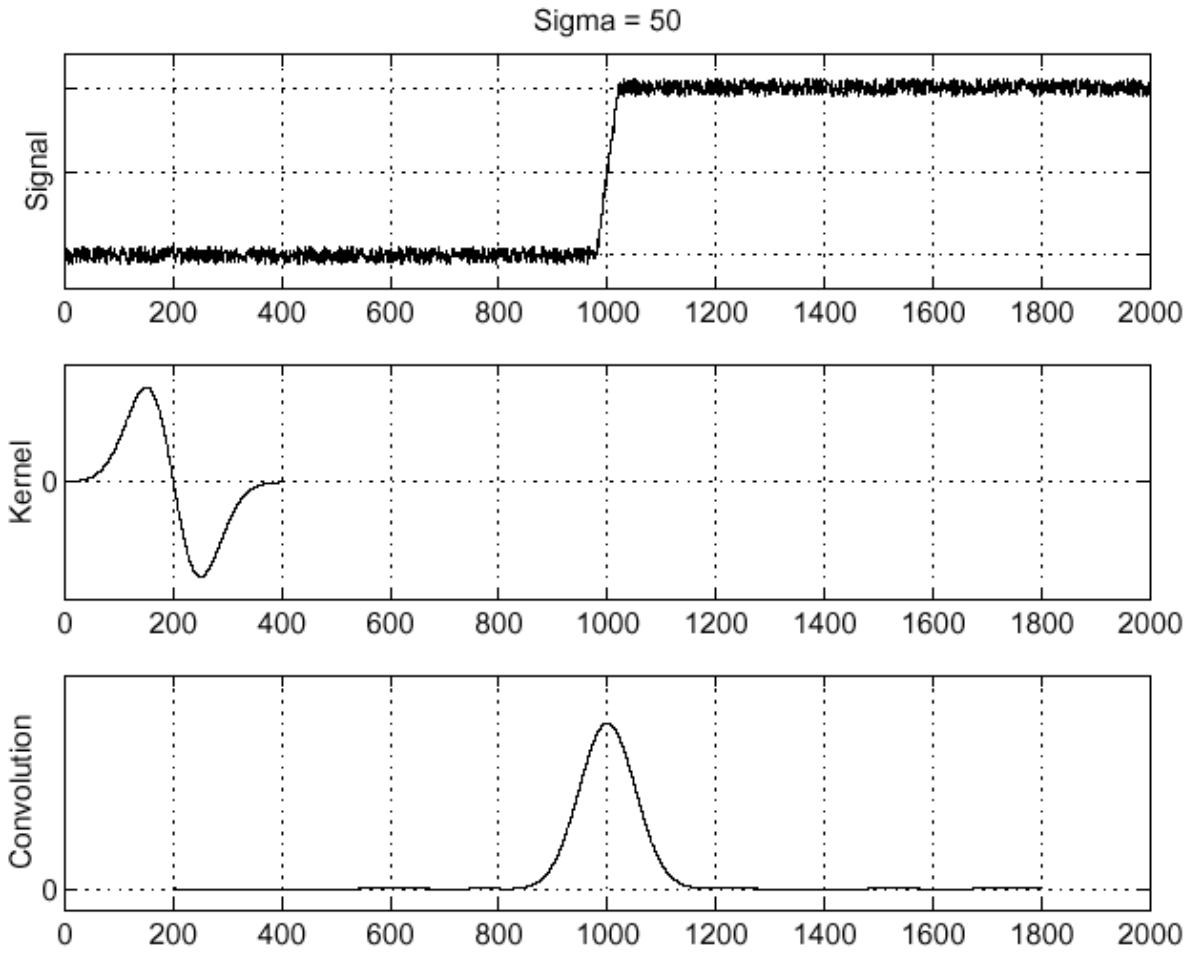
Derivative Theorem of Convolution

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

…saves us one operation.

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

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



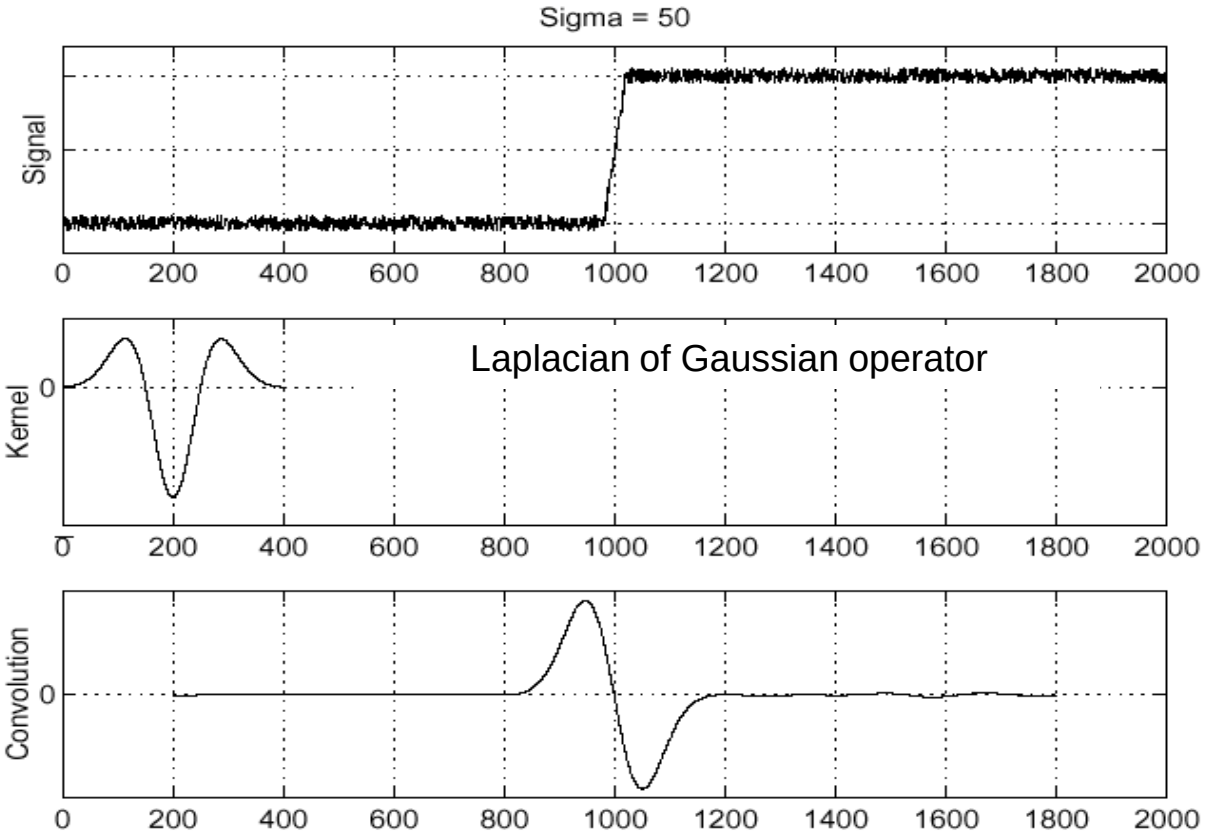
Laplacian of Gaussian (LoG)

$$\frac{\partial^2}{\partial x^2}(h * f) = \left(\frac{\partial^2}{\partial x^2} h\right) * f$$

Laplacian of Gaussian

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

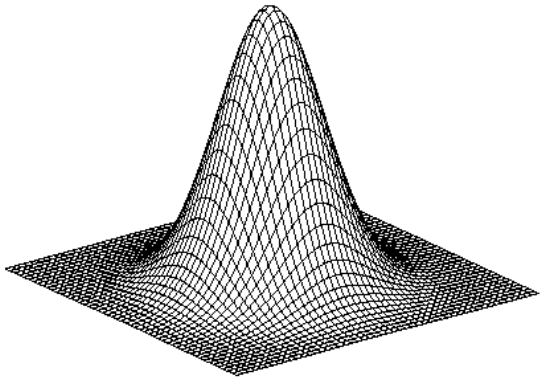
$(\frac{\partial^2}{\partial x^2} h) \star f$



Where is the edge?

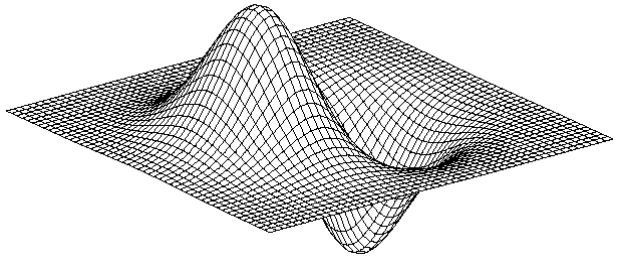
Zero-crossings of bottom graph !

2D Gaussian Edge Operators



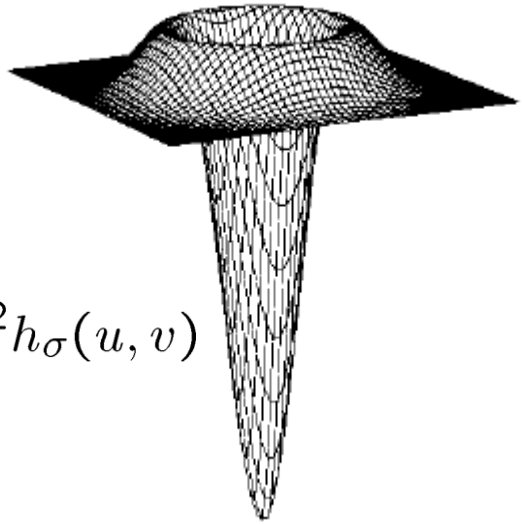
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian



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

Derivative of Gaussian (DoG)



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

Laplacian of Gaussian

Mexican Hat (Sombrero)

- ∇^2 is the **Laplacian** operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Canny Edge Operator

- Smooth image *I* with 2D Gaussian: $G * I$
- Find local edge normal directions for each pixel

$$\bar{\mathbf{n}} = \frac{\nabla (G * I)}{\left| \nabla (G * I) \right|}$$

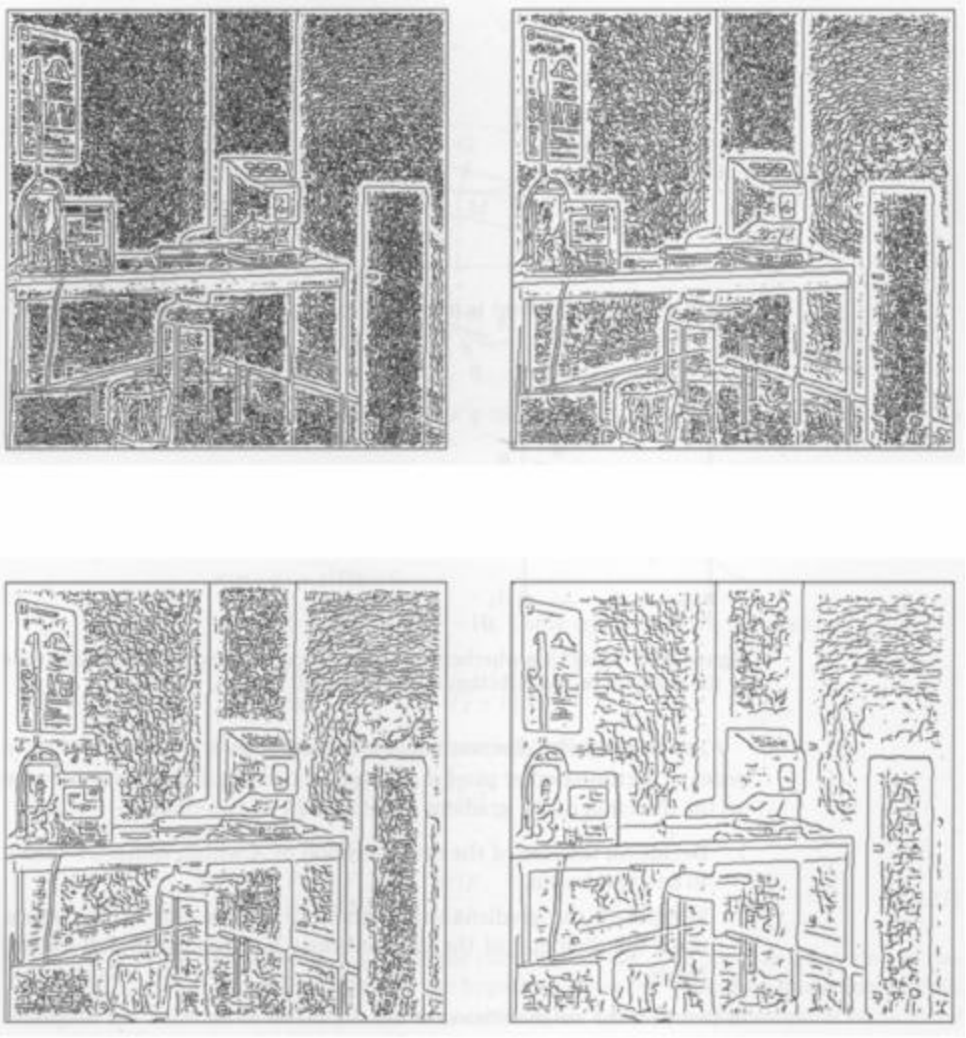
- Compute edge magnitudes $\left| \nabla (G * I) \right|$
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \bar{\mathbf{n}}^2} = 0$$

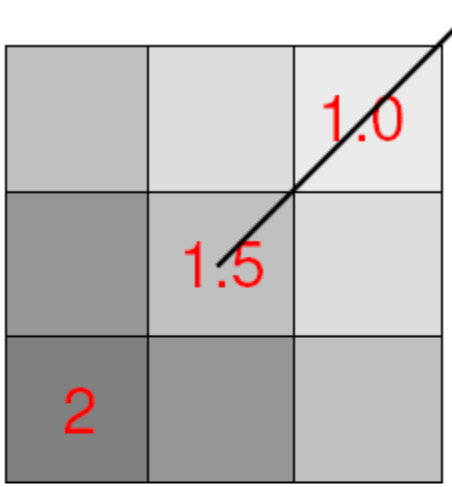
Canny Edge Operator

Canny Edge Operator

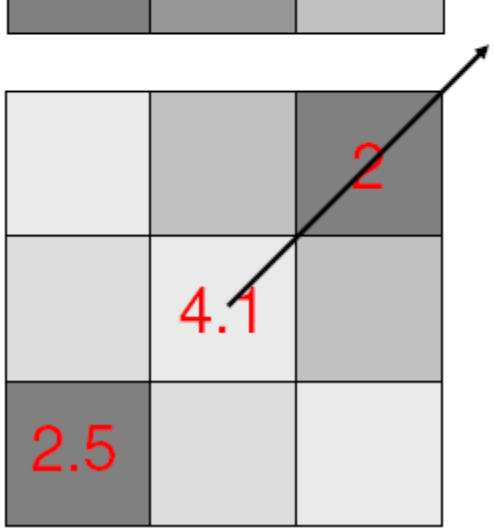
Different thresholds applied to gradient magnitude



Non-local maxima suppression

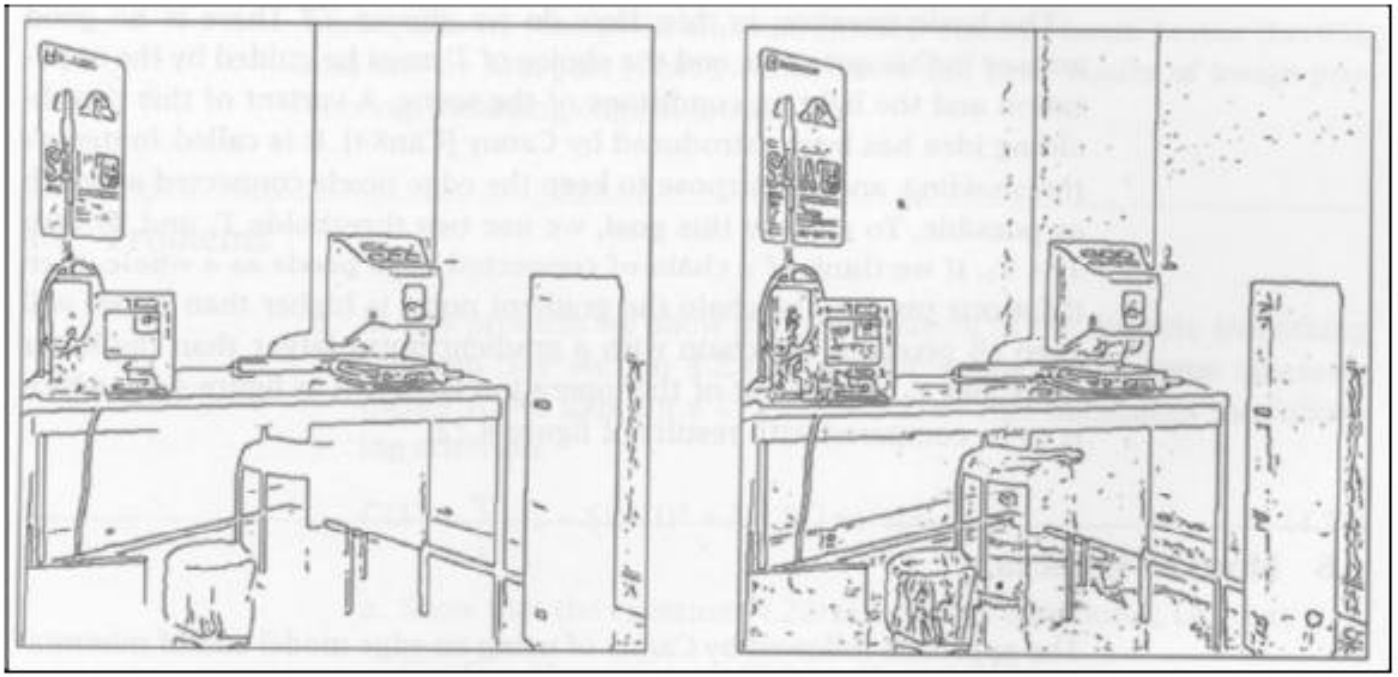


Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)



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

Non-local maxima suppression

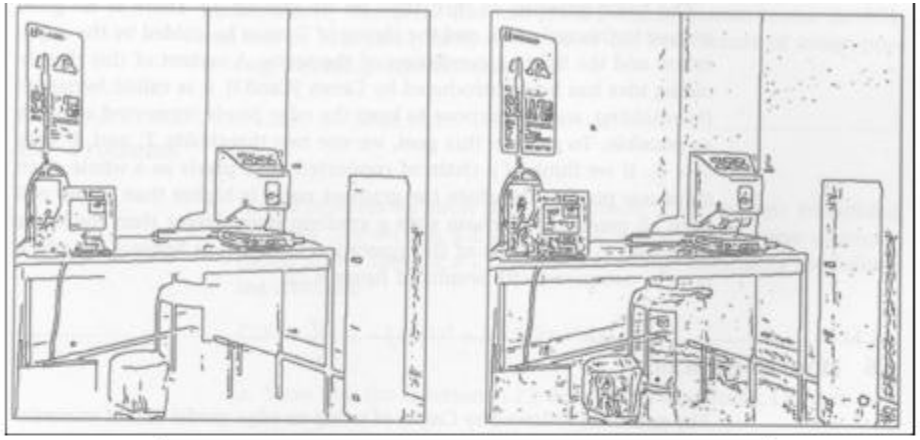


T = 15 T = 5

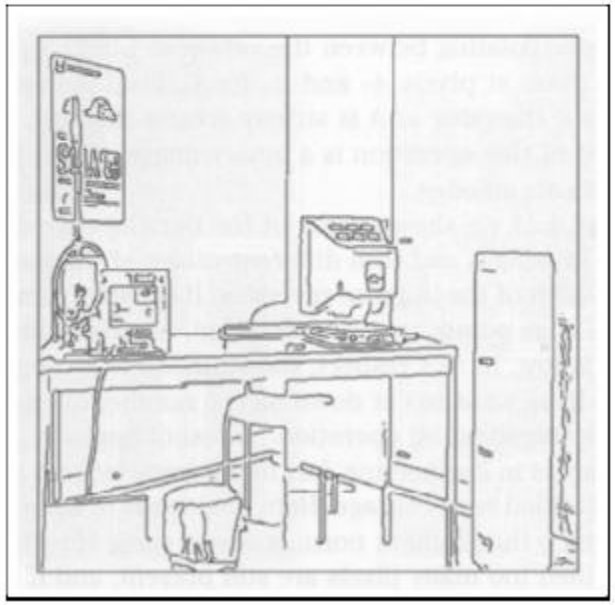
Two thresholds applied to gradient magnitude

Hysteresis thresholding

T=15



T=5



Hysteresis
T_h=15 T_l = 5

Hysteresis
thresholding

Resources

- Russ – Chapter 8, 9
- Gonzalez & Woods – Chapter 4
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm