VC 12/13 – T7
Spatial Filters

Mestrado em Ciência de Computadores

Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos

Miguel Tavares Coimbra

Outline

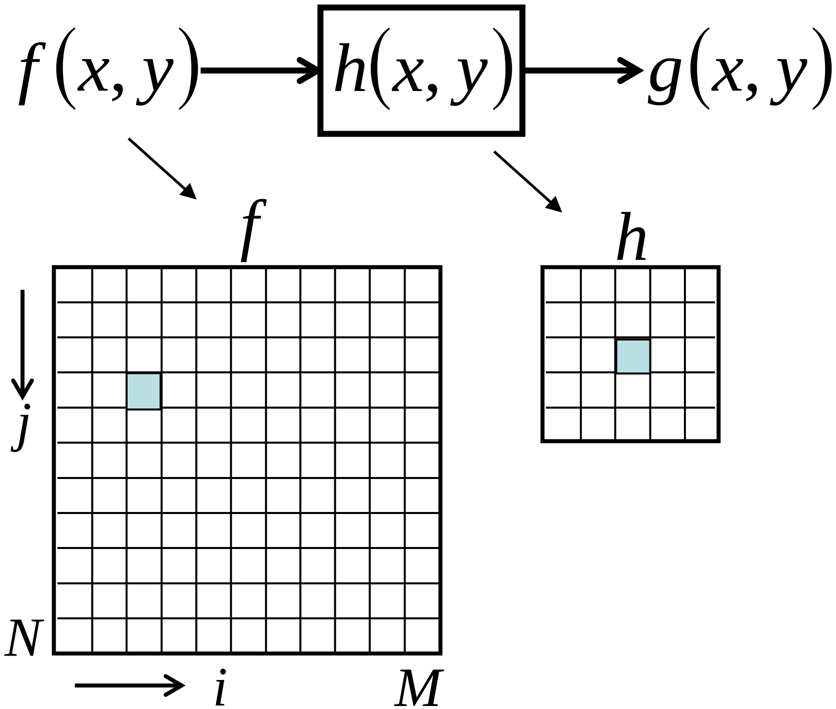
- Spatial filters
- Frequency domain filtering
- Edge detection

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. Please acknowledge the original source when reusing these slides for academic purposes.

Topic: Spatial filters

- 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

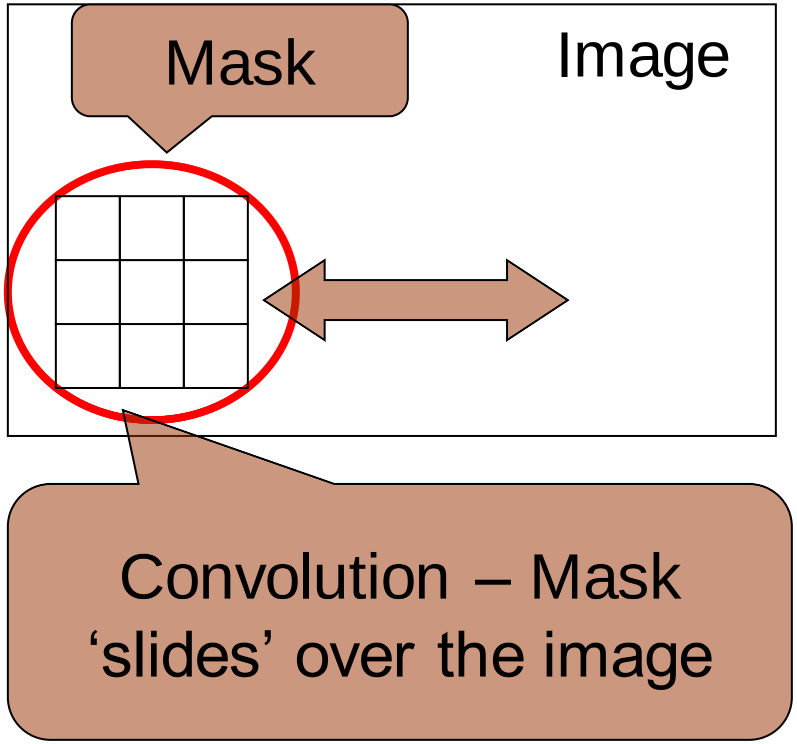
$$F(u,v)=\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)\,e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

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.
- Corresponds to a multiplication in frequency domain.



Example

- 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

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

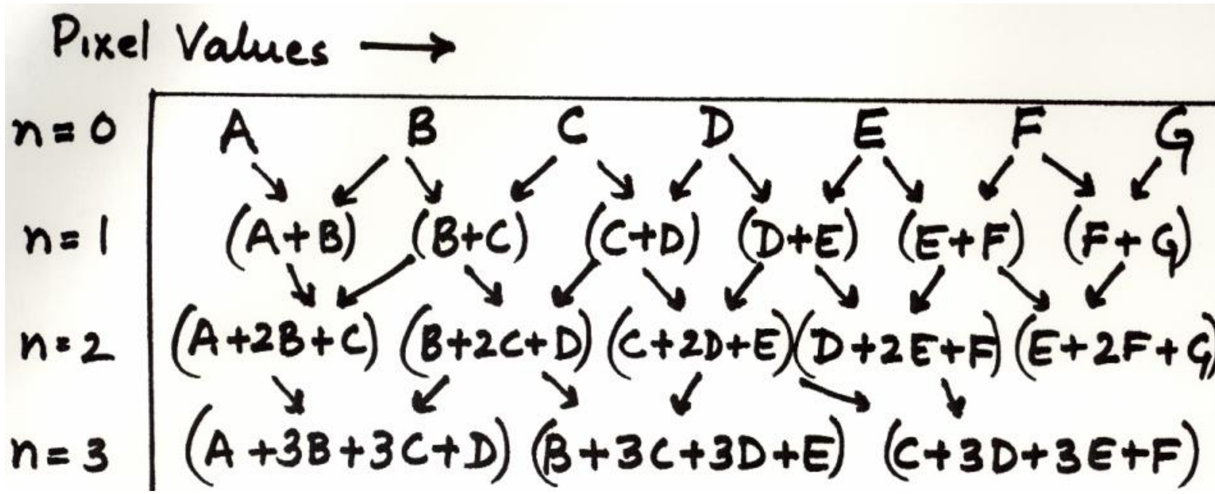
=1*2+2*2+1*2+…
=8+0-20
=-12

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**.

Averaging

Let's think about averaging pixel values



For *n=2*, convolve pixel values with $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$

Which is faster?

$(a)\,O(2(n+1))$ $\quad (b)\,O\left((n+1)^2\right)$

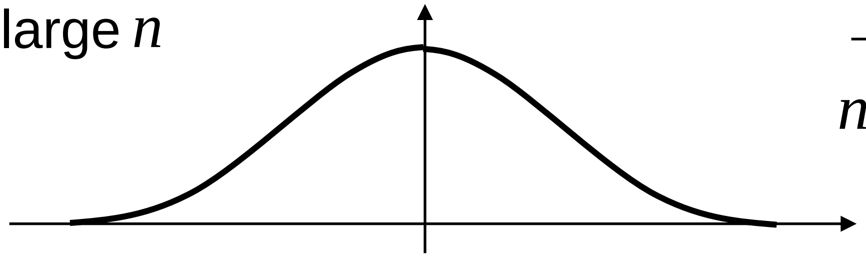
2D images:

(a) use $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$ then $\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array}$ or (b) use $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array} * \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$

Averaging

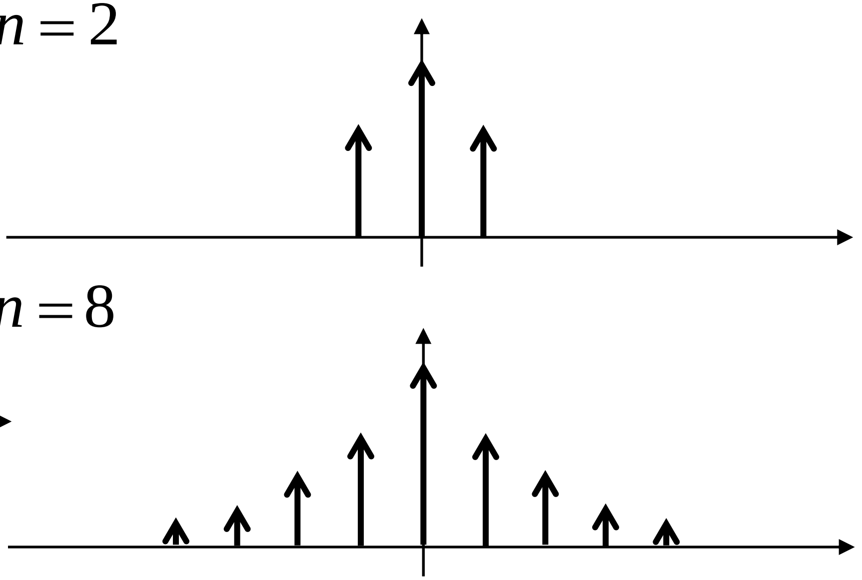
The convolution kernel

large n



$n = 2$

$n = 8$

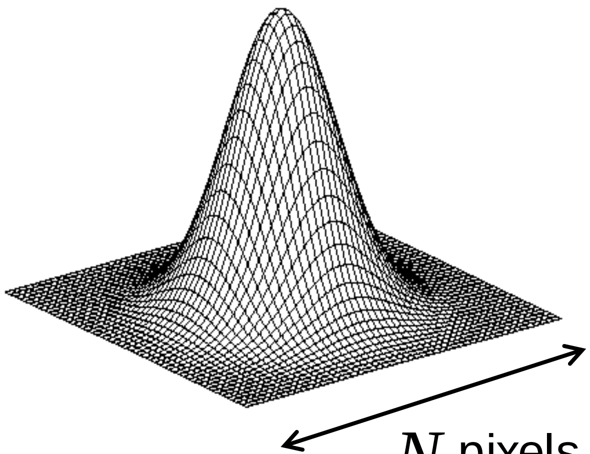


Repeated averaging $\approx$ Gaussian smoothing

Gaussian Smoothing

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!

U.PORTO FC

Gaussian Smoothing

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

$H[u,v]$

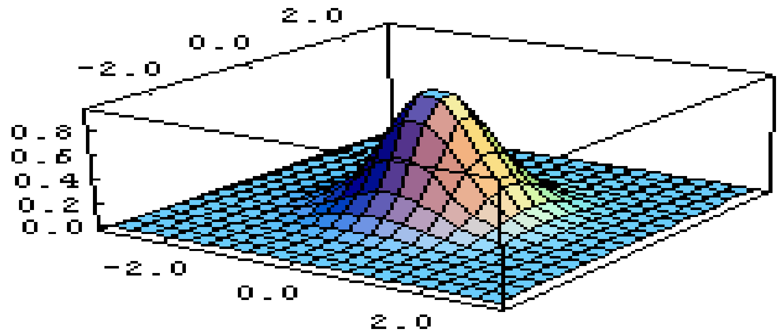
$$\frac{1}{16}$$

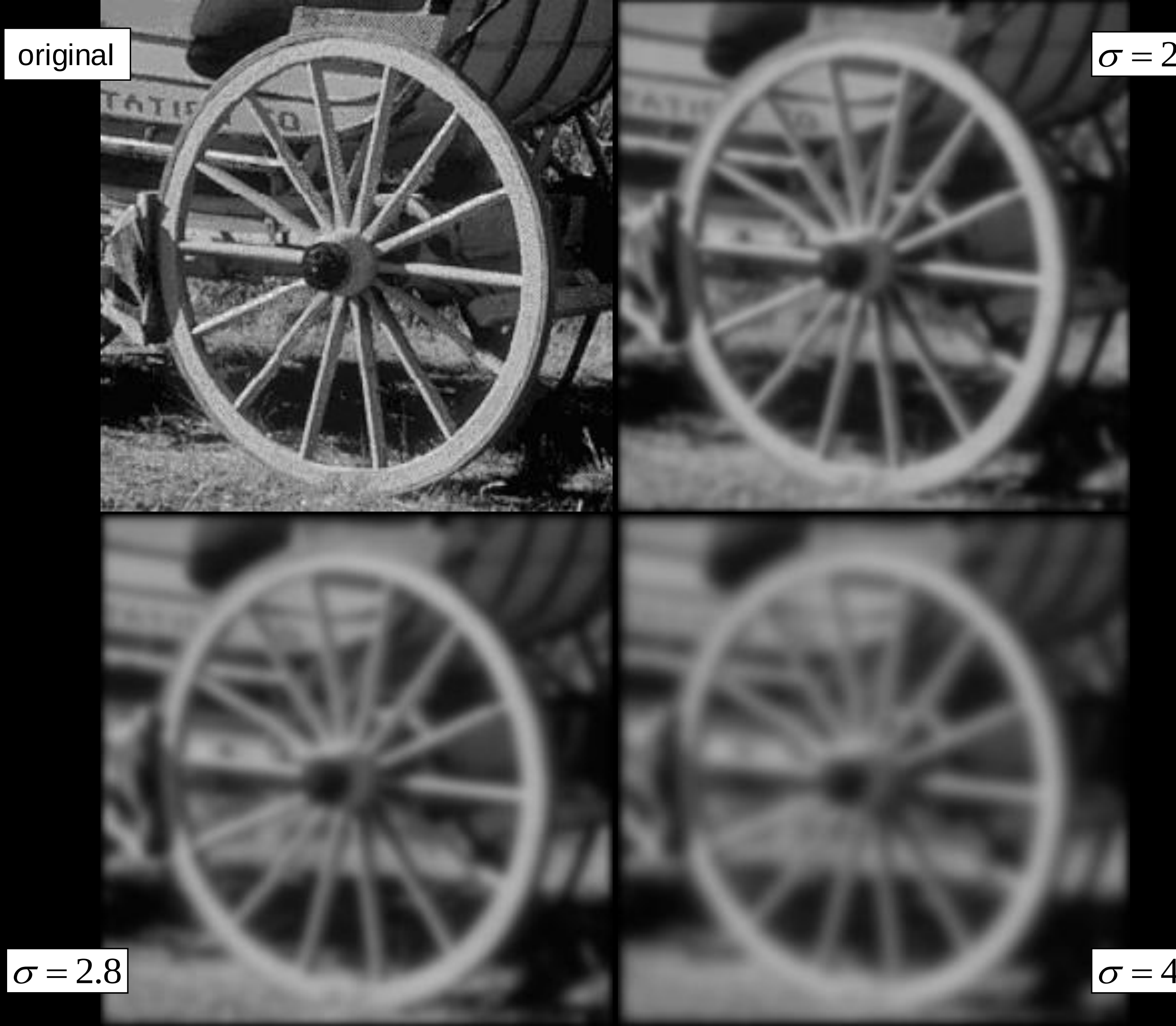
1	2	1
2	4	2
1	2	1

- 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
$\sigma = 2$
$\sigma = 2.8$
$\sigma = 4$

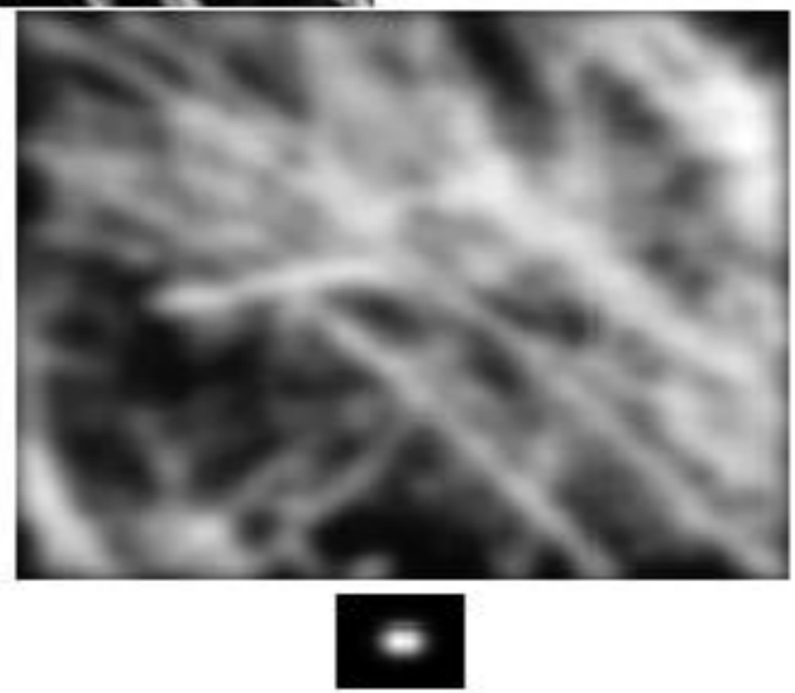
Mean Filtering

- 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

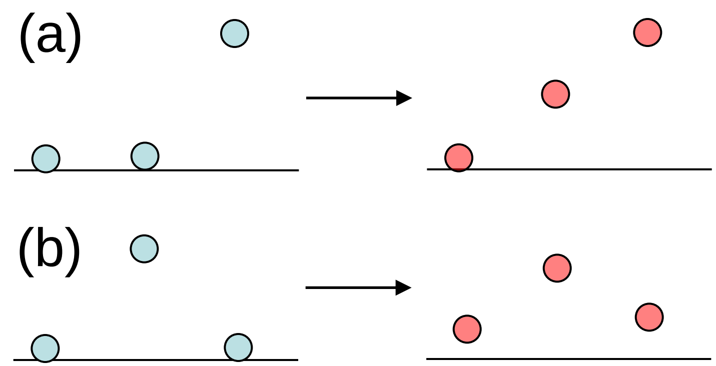


Gaussian filter

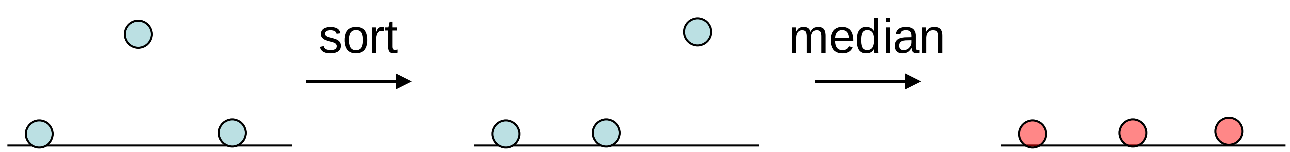
http://www.michaelbach.de/ot/cog_blureffects/index.html

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**

Border Problem

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

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

How do we apply our mask to this pixel?

What a computer sees

Border Problem

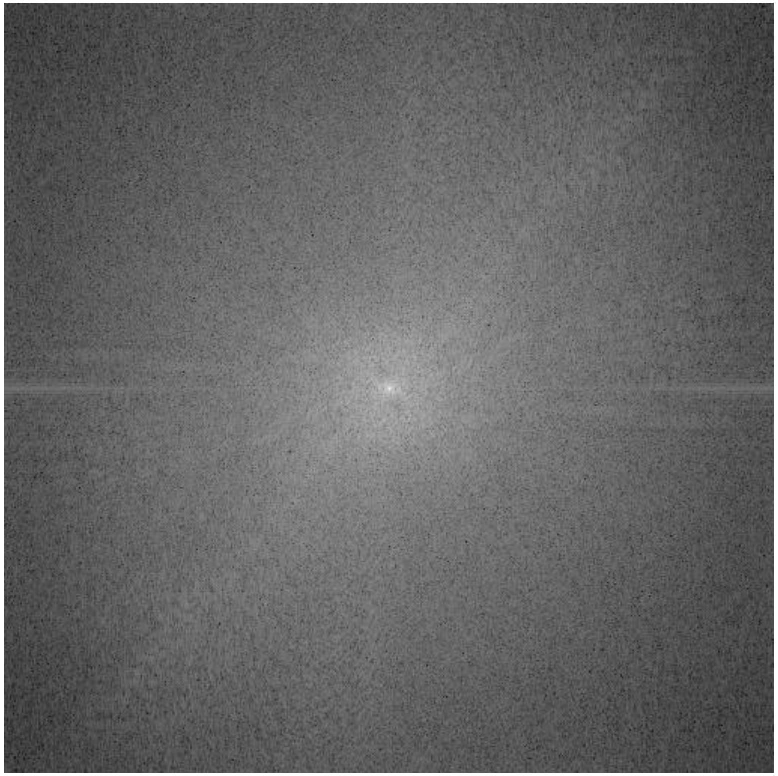
- Ignore
  - Output image will be smaller than original
- Pad with constant values
  - Can introduce substantial 1^{st} order derivative values
- Pad with reflection
  - Can introduce substantial 2^{nd} order derivative values

Topic: Frequency domain filtering

- Spatial filters
- Frequency domain filtering
- Edge detection

Image Processing in the Fourier Domain

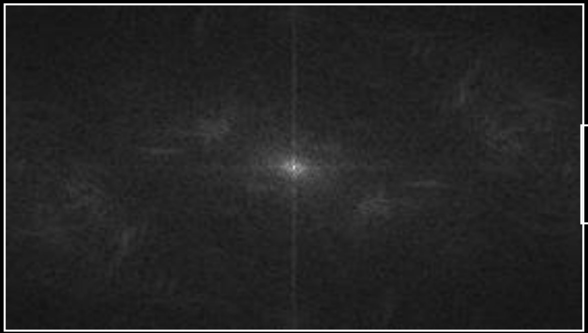
Magnitude of the FT



Does not look anything like what we have seen

Convolution in the Frequency Domain

$f(x,y)$



$|F(s_x,s_y)|$

$h(x,y)$

$|H(s_x,s_y)|$

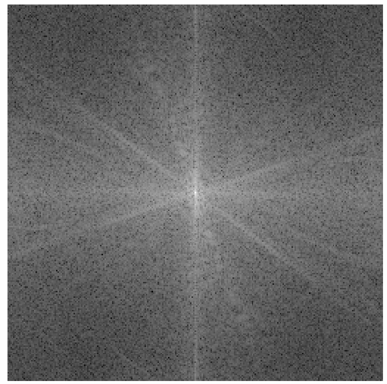
$g(x,y)$

$|G(s_x,s_y)|$

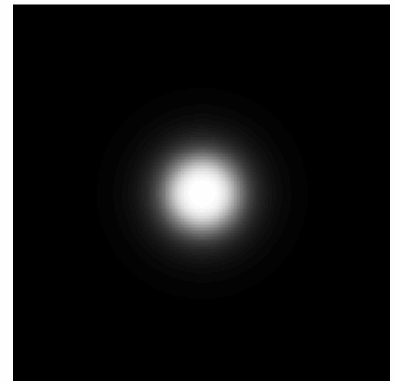
Low-pass Filtering

Original image

FFT of original image

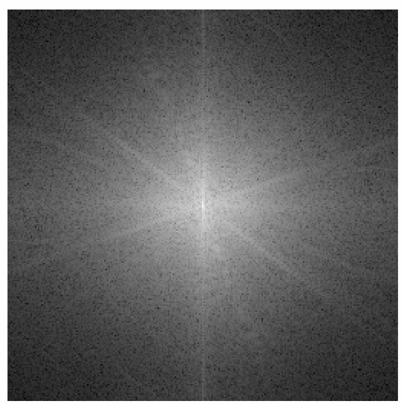


Low–pass filter



Low–pass image

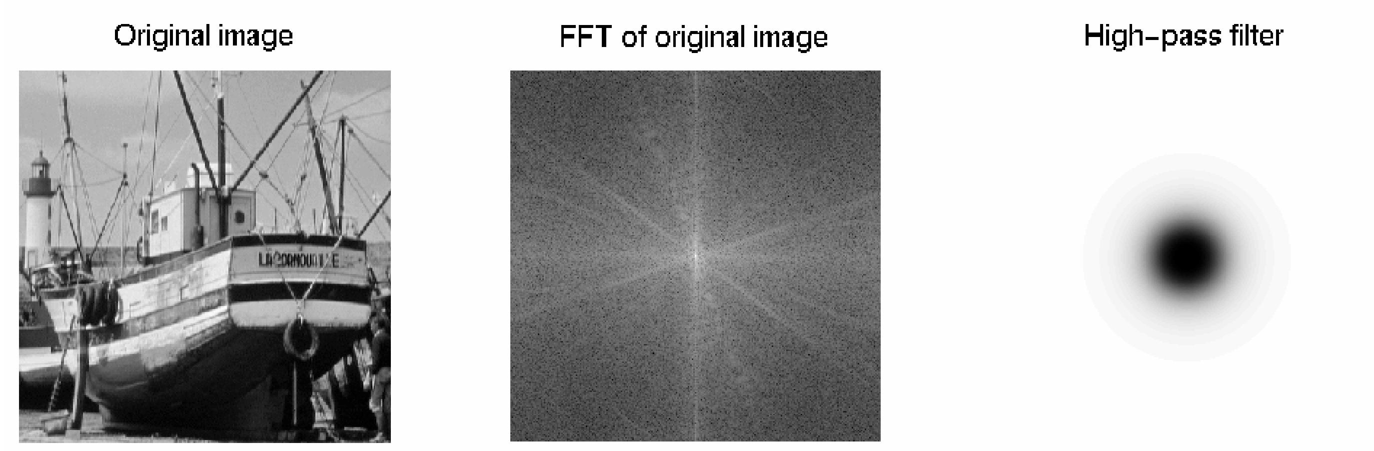
FFT of low–pass image

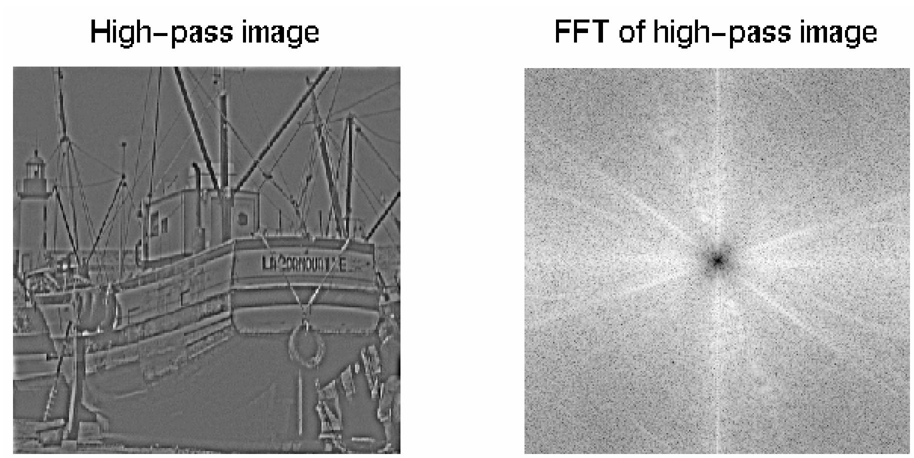


Lets the low frequencies pass and eliminates the high frequencies.

Generates image with overall shading, but not much detail

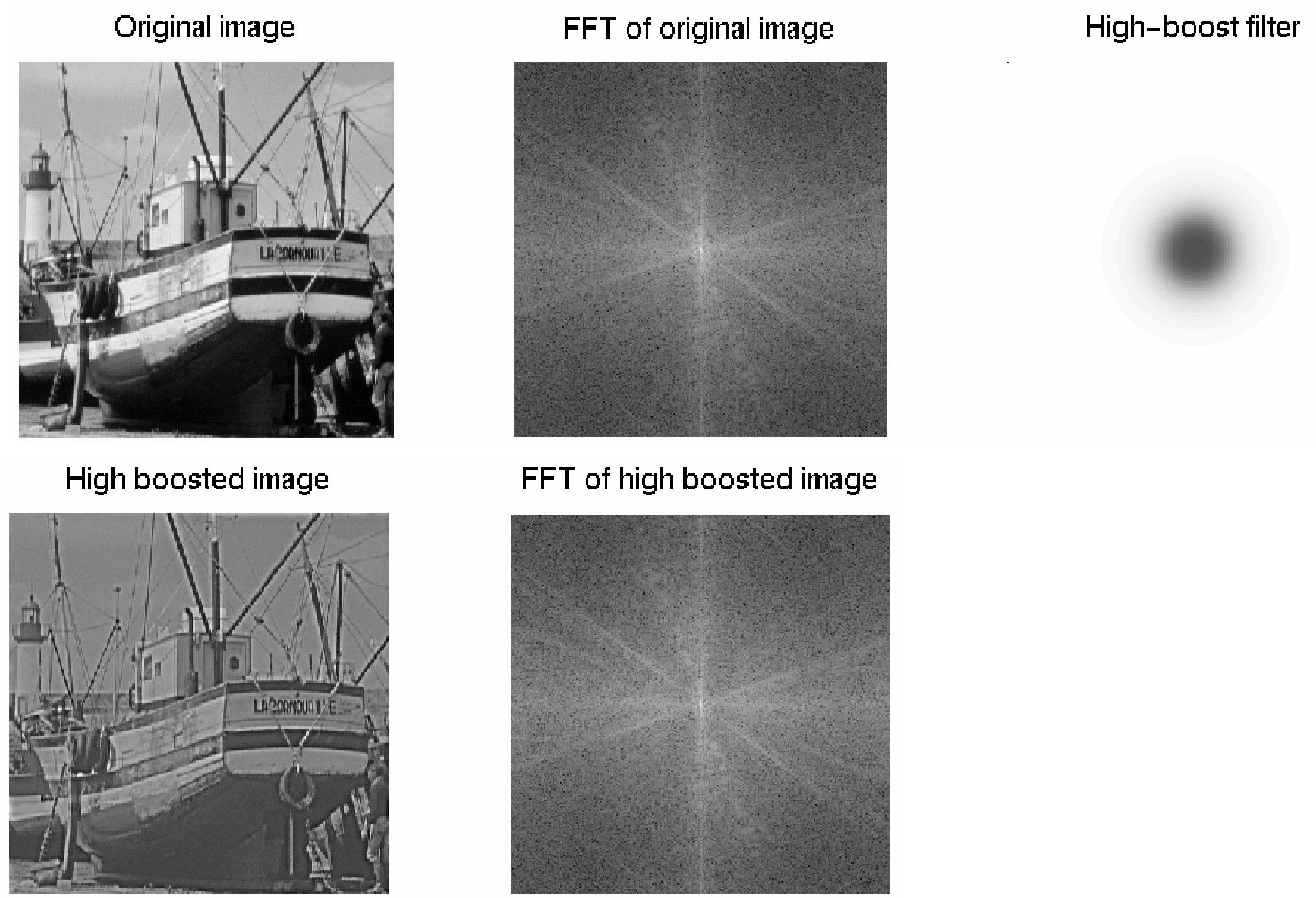
High-pass Filtering

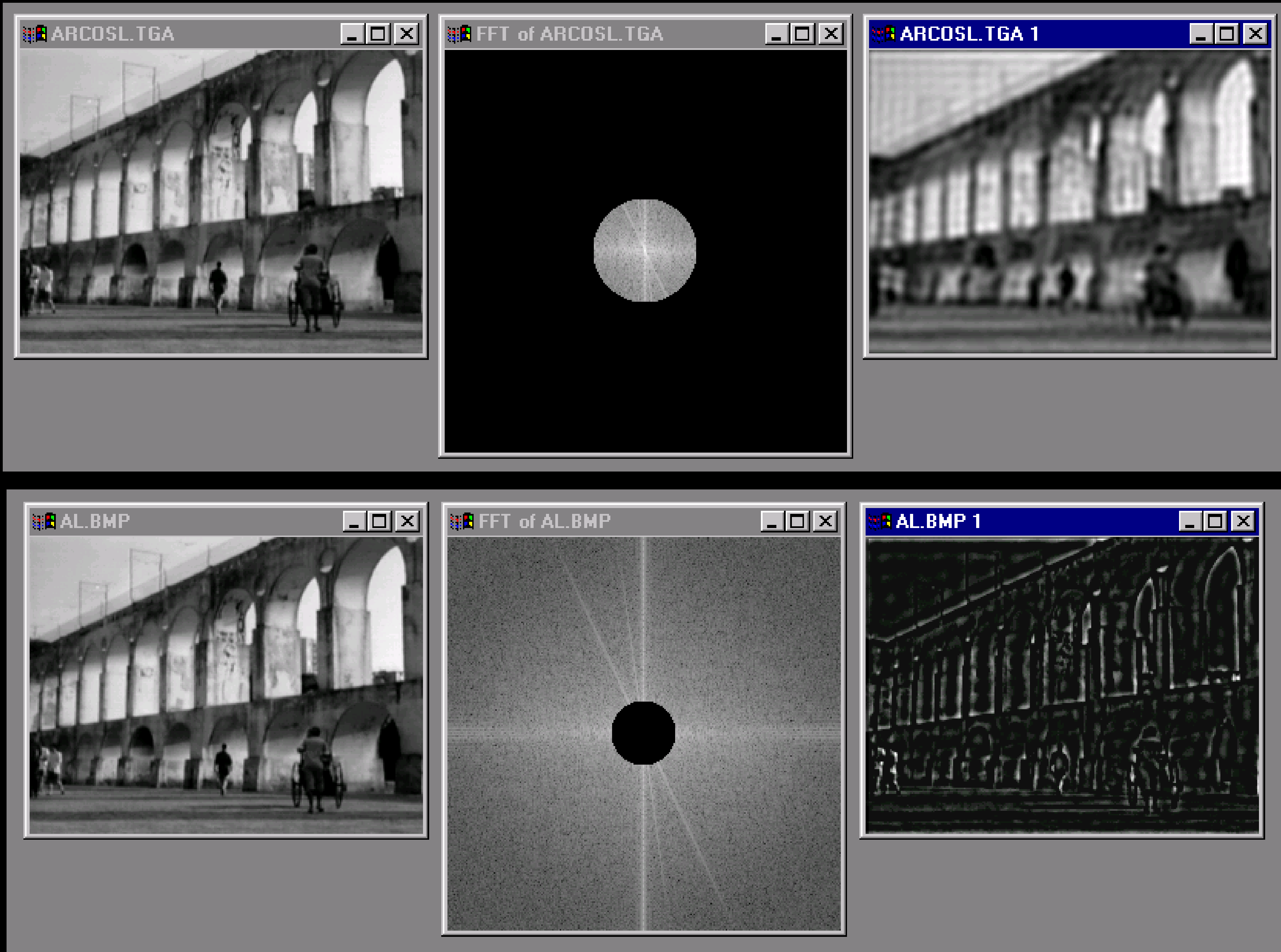




Lets through the high frequencies (the detail), but eliminates the low frequencies (the overall shape). It acts like an edge enhancer.

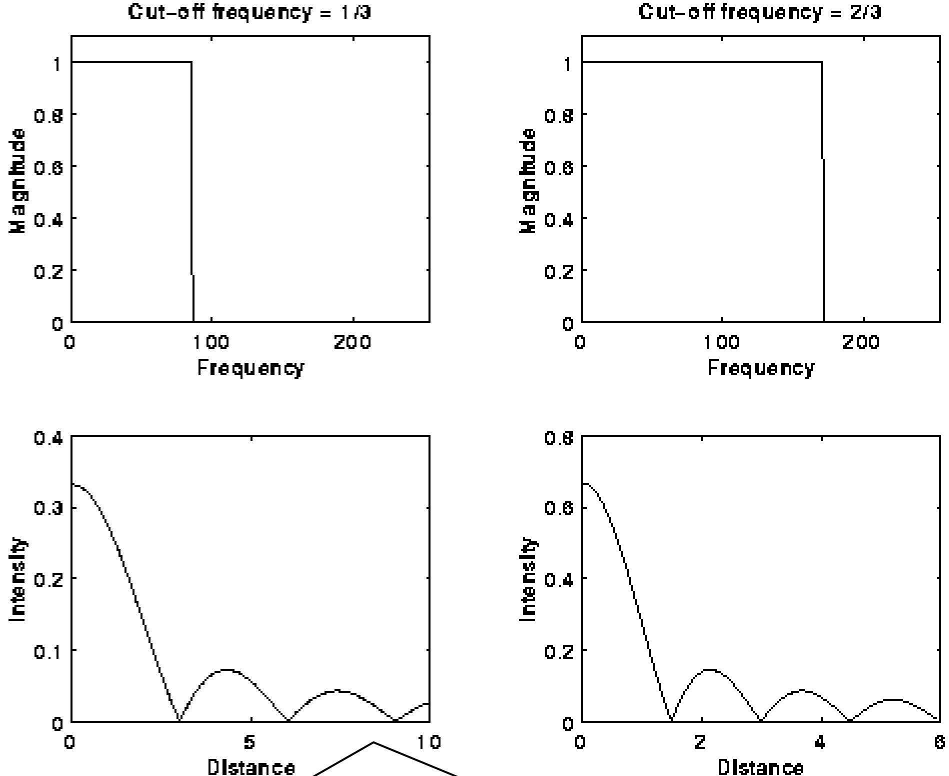
Boosting High Frequencies




ARCOSL.TGA
FFT of ARCOSL.TGA
ARCOSL.TGA 1
AL.BMP
FFT of AL.BMP
AL.BMP 1


FFT of ARCOSL.TGA
ARCOSL.TGA 1
FFT of ARCOSL.TGA
ARCOSL.TGA 1
FFT of ARCOSL.TGA
ARCOSL.TGA 1
FFT of ARCOSL.TGA
ARCOSL.TGA 1

The Ringing Effect



An ideal low-pass filter causes 'rings' in the spatial domain!

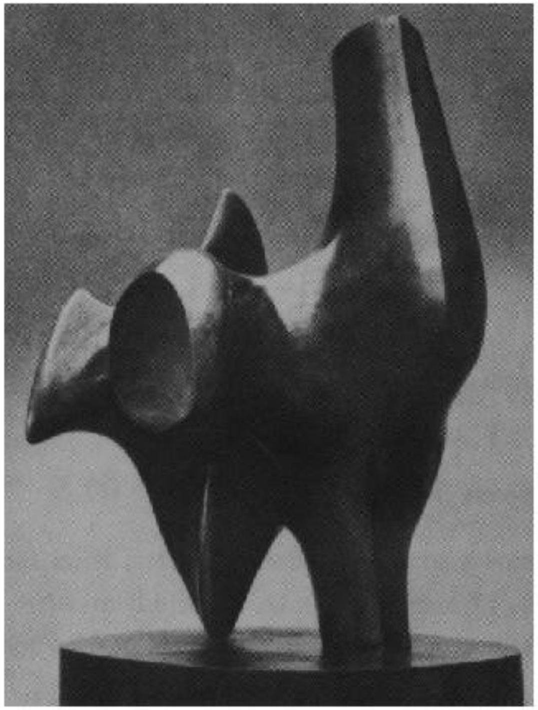
http://homepages.inf.ed.ac.uk/rbf/HIPR2/freqfilt.htm

Topic: Edge detection

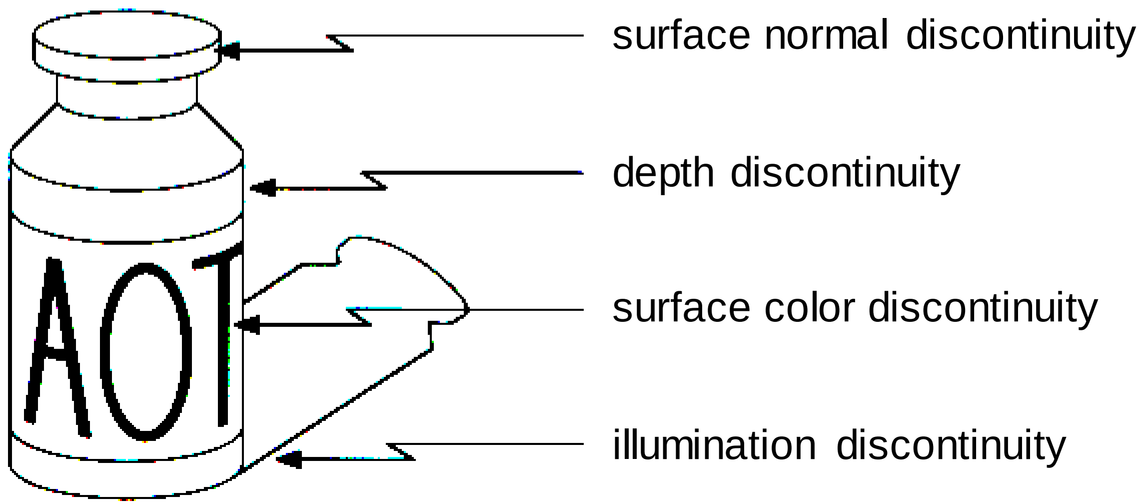
- Spatial filters
- Frequency domain filtering
- Edge detection

Edge Detection

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

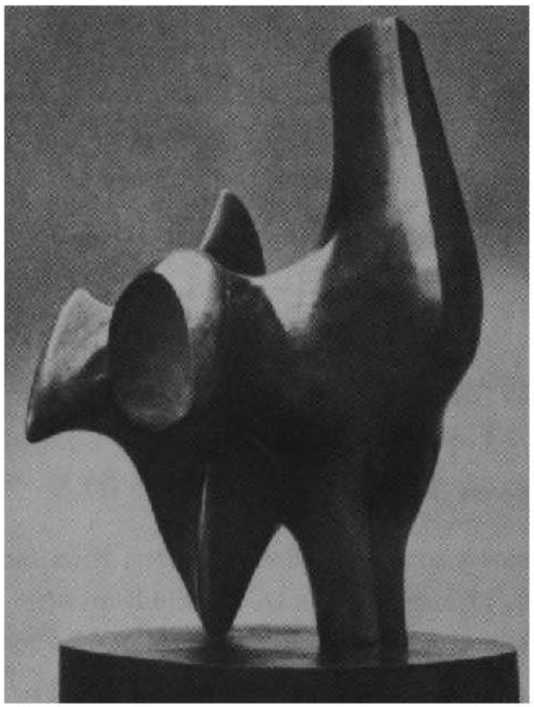


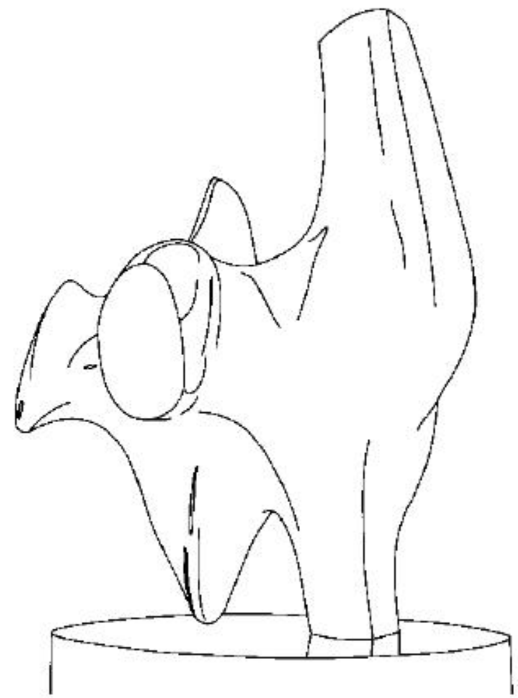
Origin of Edges



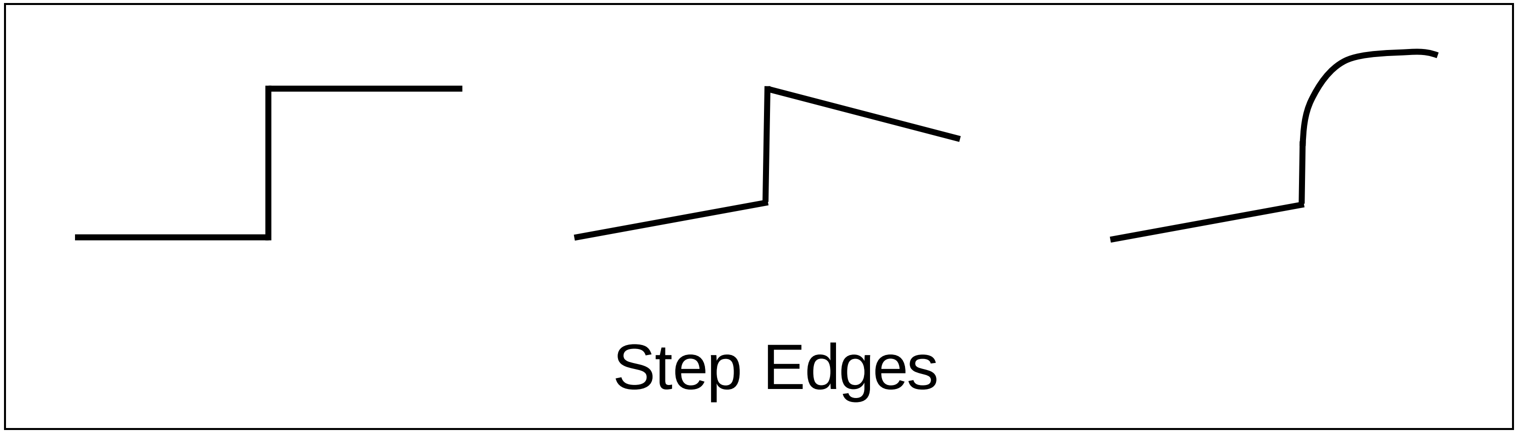
- Edges are caused by a variety of factors

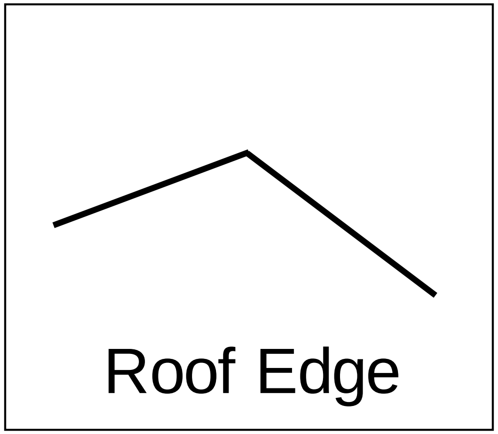
How can you tell that a pixel is on an edge?

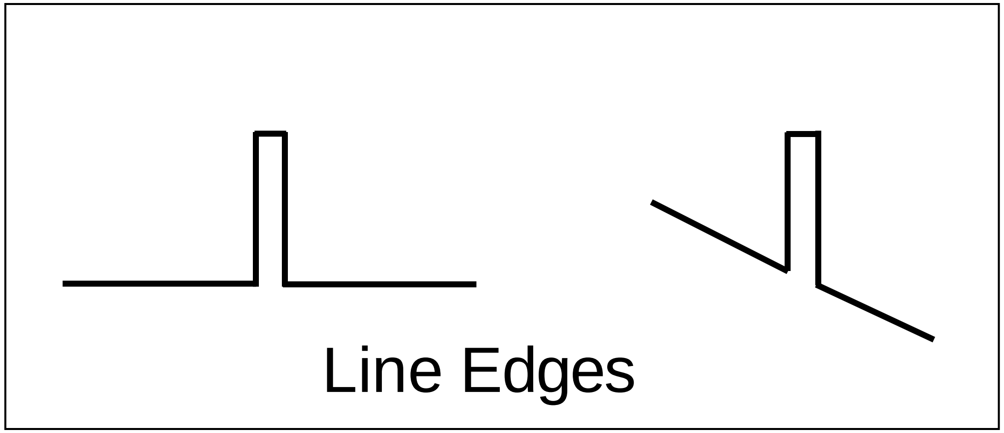




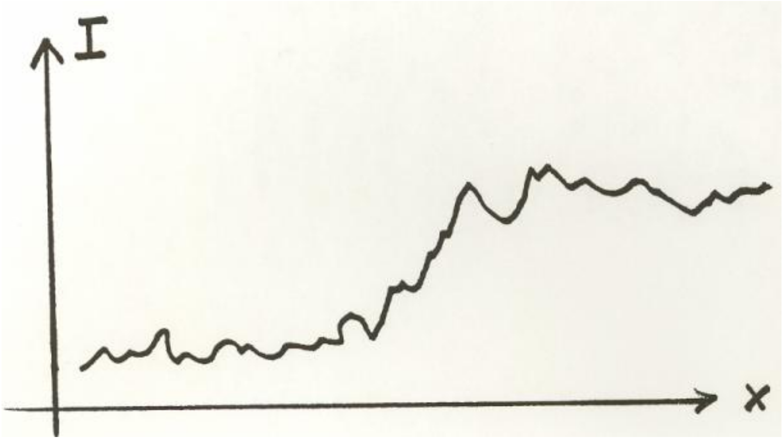
Edge Types







Real Edges



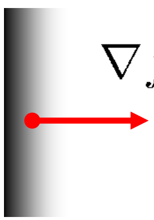
Noisy and Discrete!

We want an **Edge Operator** that produces:

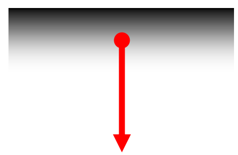
- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

Gradient

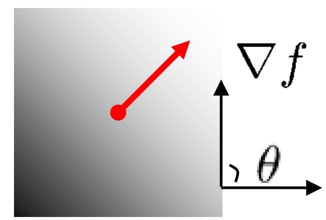
- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity



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



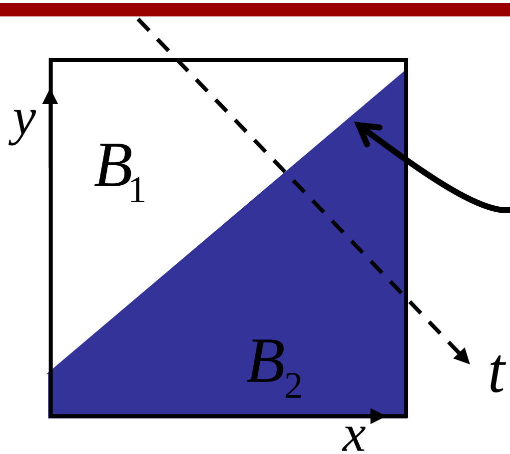
$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



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

- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

Theory of Edge Detection



Ideal edge

$$L(x,y) = x\sin\theta - y\cos\theta + \rho = 0$$

$$B_1 : L(x,y) < 0$$

$$B_2 : L(x,y) > 0$$

Unit step function:

$$u(t) = \begin{cases} 1 & \text{for } t > 0 \\ \tfrac{1}{2} & \text{for } t = 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad u(t) = \int_{-\infty}^{t} \delta(s)\, ds$$

Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)\, u(x\sin\theta - y\cos\theta + \rho)$$

Theory of Edge Detection

- Partial derivatives (gradients):

$$\frac{\partial I}{\partial x} = +\sin\theta (B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos\theta (B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

- Squared gradient:

$$s(x,y) = \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2 = \left[(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)\right]^2$$

Edge Magnitude: $\sqrt{s(x,y)}$

Edge Orientation: $\arctan\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$ (normal of the edge)

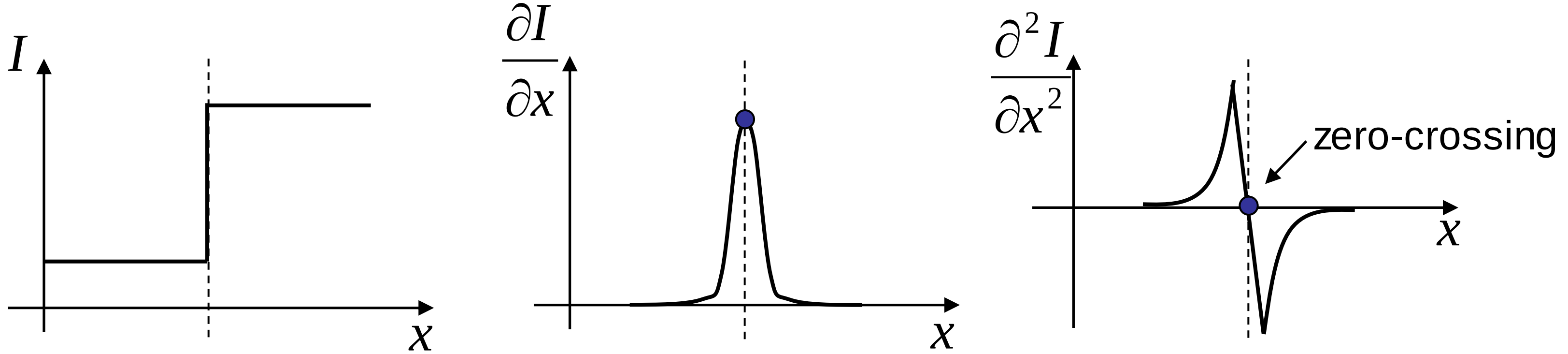
Rotationally symmetric, non-linear operator

Theory of Edge Detection

- Laplacian:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \left(B_2 - B_1\right)\delta'\left(x\sin\theta - y\cos\theta + \rho\right)$$

Rotationally symmetric, linear operator



Discrete Edge Operators

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

Finite difference approximations:

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

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

$$\begin{array}{|c|c|} \hline I_{i,j+1} & I_{i+1,j+1} \\ \hline I_{i,j} & I_{i+1,j} \\ \hline \end{array} \quad \varepsilon$$

Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array}$$

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 :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array} \quad \text{or} \quad \frac{1}{6\varepsilon^2}\begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array}$$

(more accurate)

The Sobel Operators

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

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

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

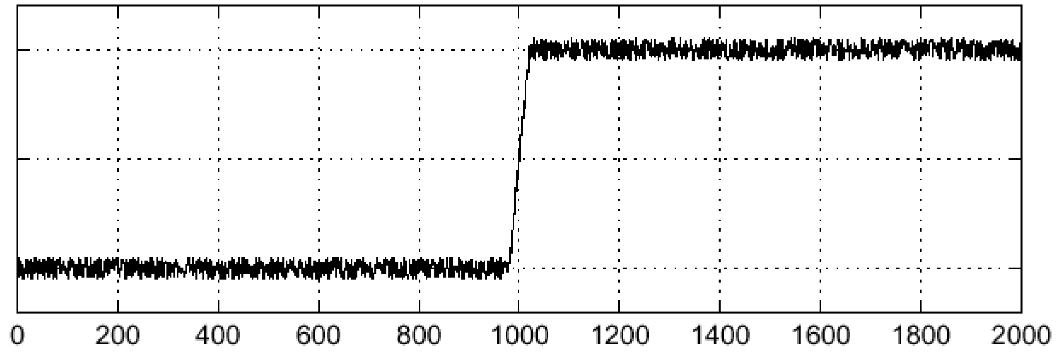
1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

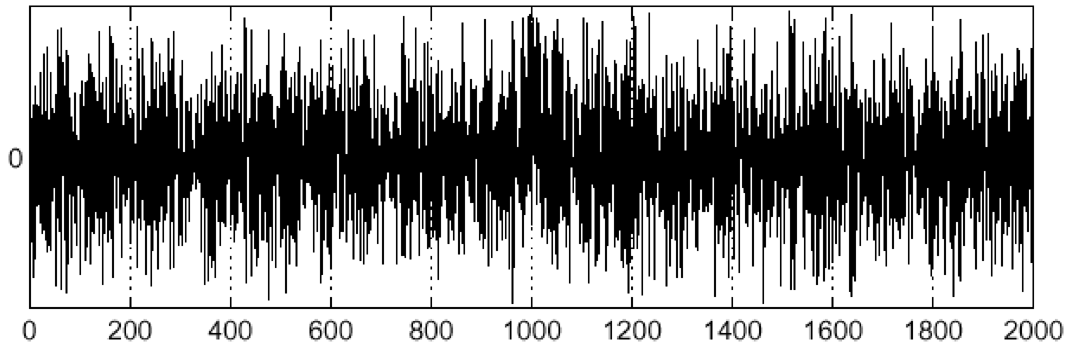
Effects of Noise

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

$f(x)$

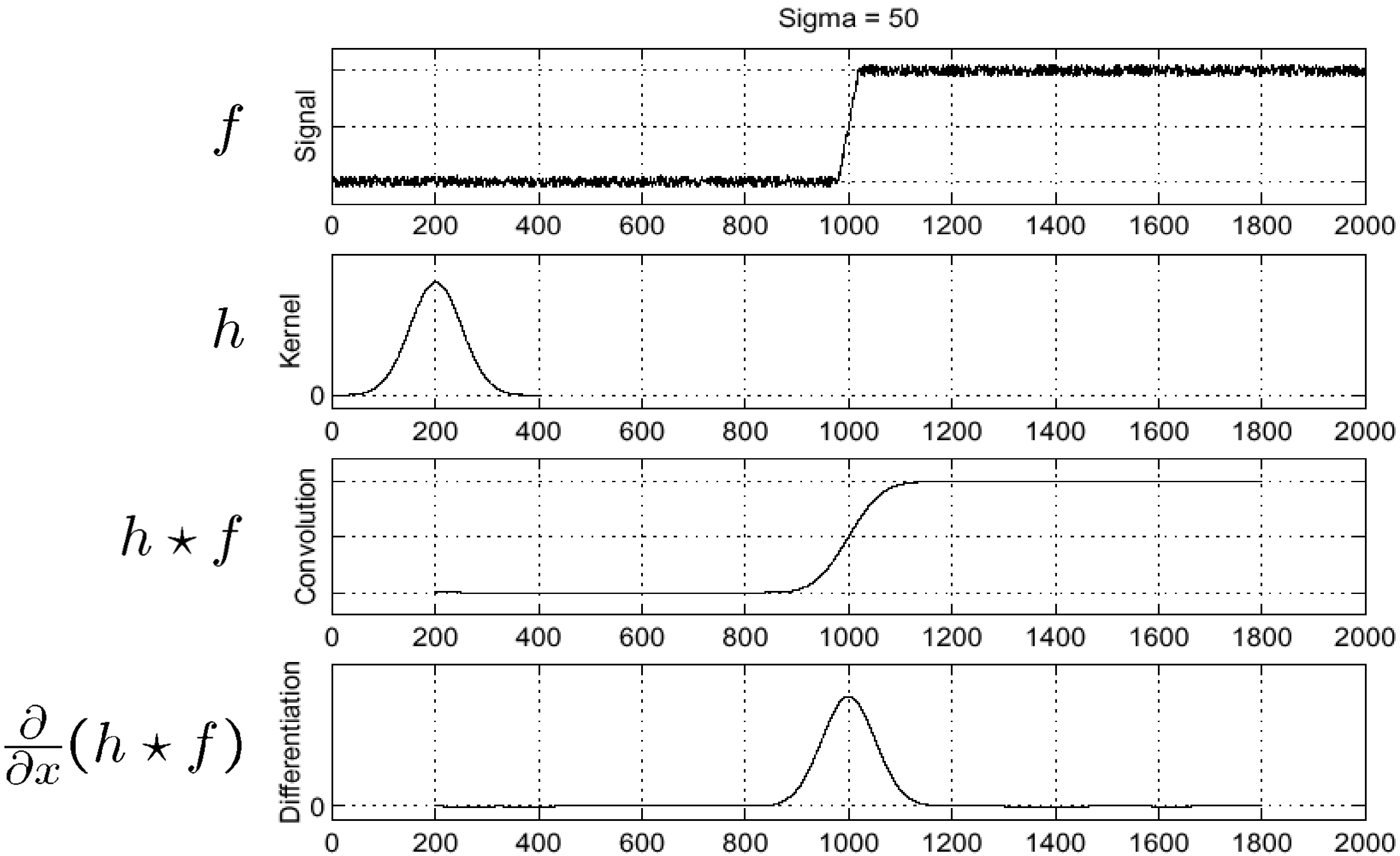


$\frac{d}{dx}f(x)$



Where is the edge??

Solution: Smooth First



Where is the edge?

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

Derivative Theorem of Convolution

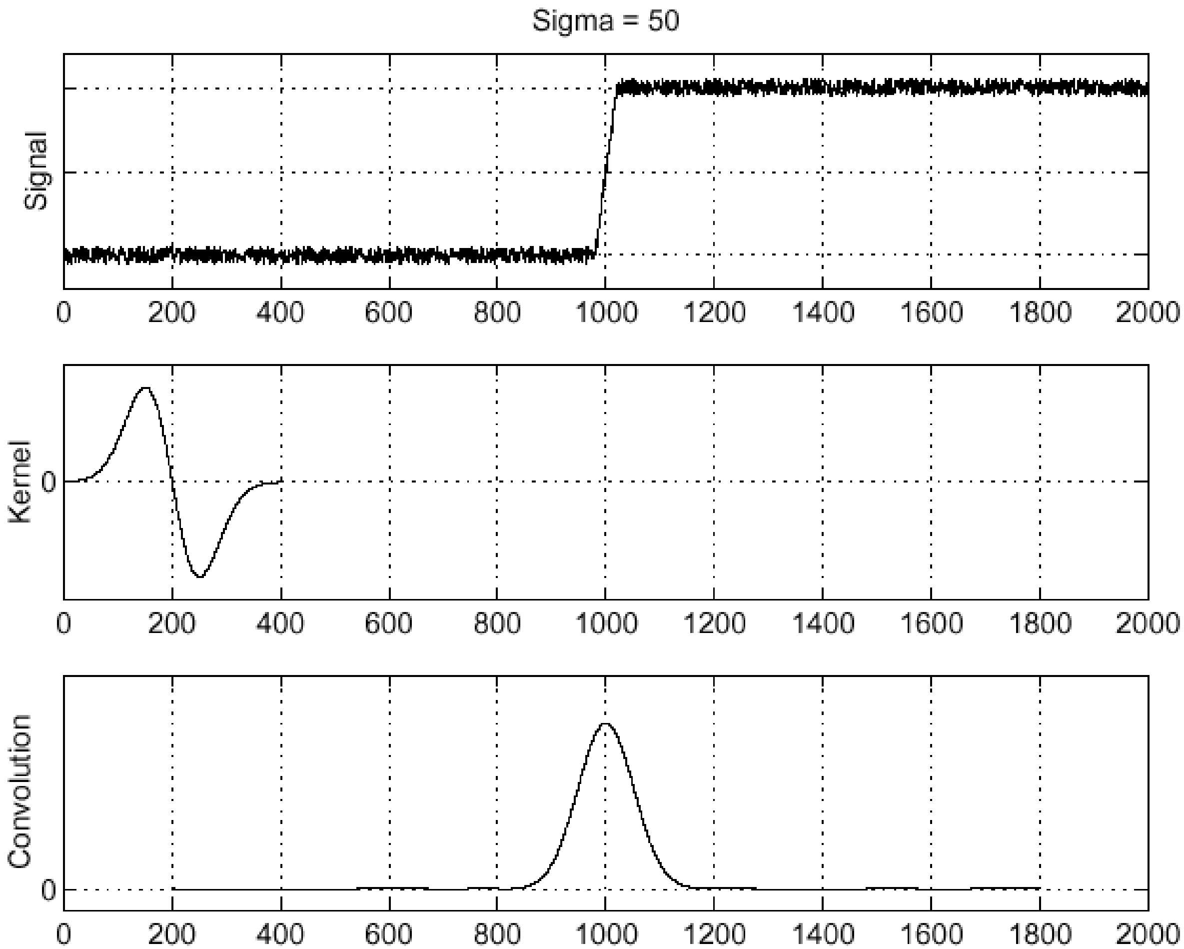
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

…saves us one operation.

f

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

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



Laplacian of Gaussian (LoG)

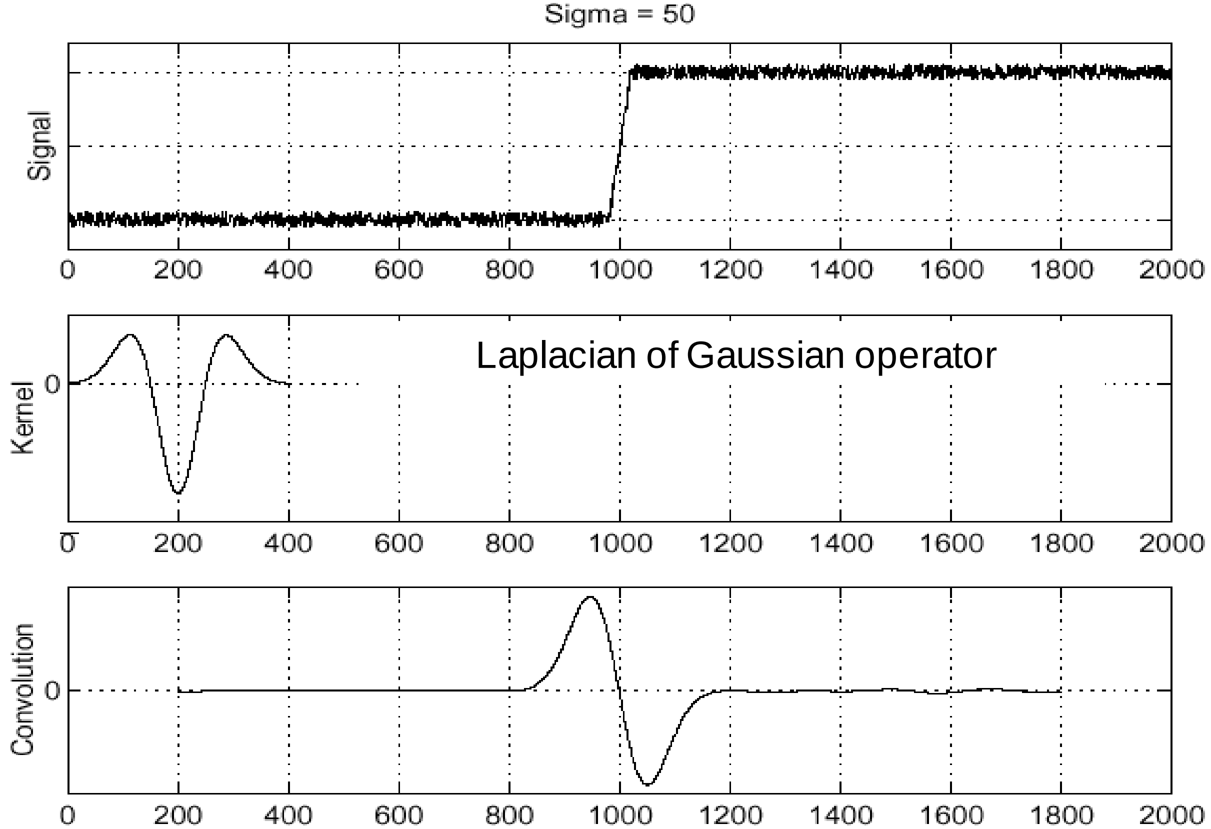
$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2} h\right)} * f$$

Laplacian of Gaussian

f

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

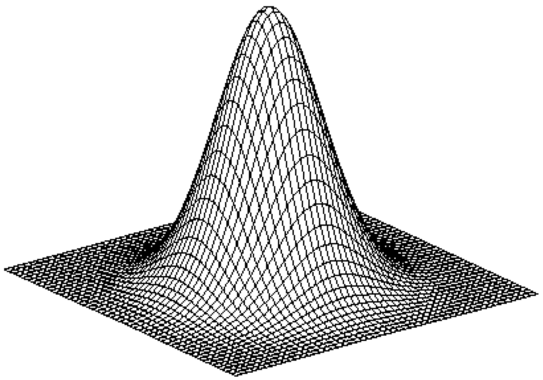
$(\frac{\partial^2}{\partial x^2}h) \star f$

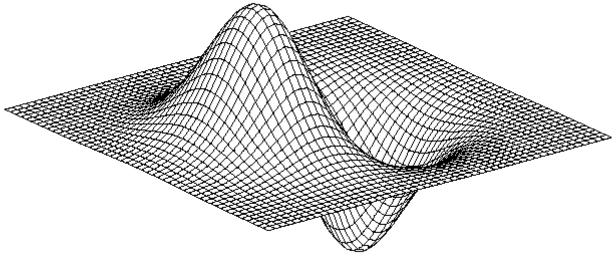


Where is the edge?

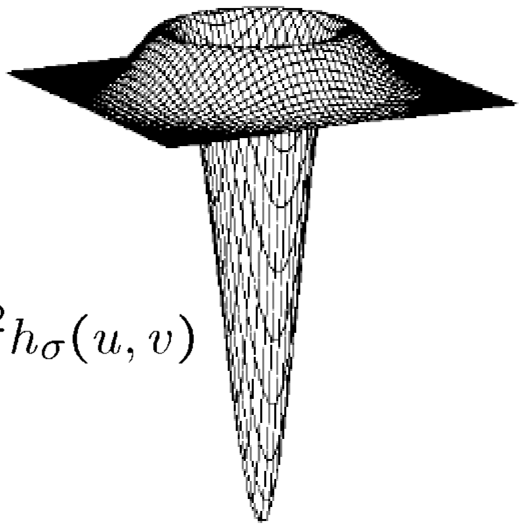
Zero-crossings of bottom graph !

2D Gaussian Edge Operators





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



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

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

Gaussian

Derivative of Gaussian (DoG)

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

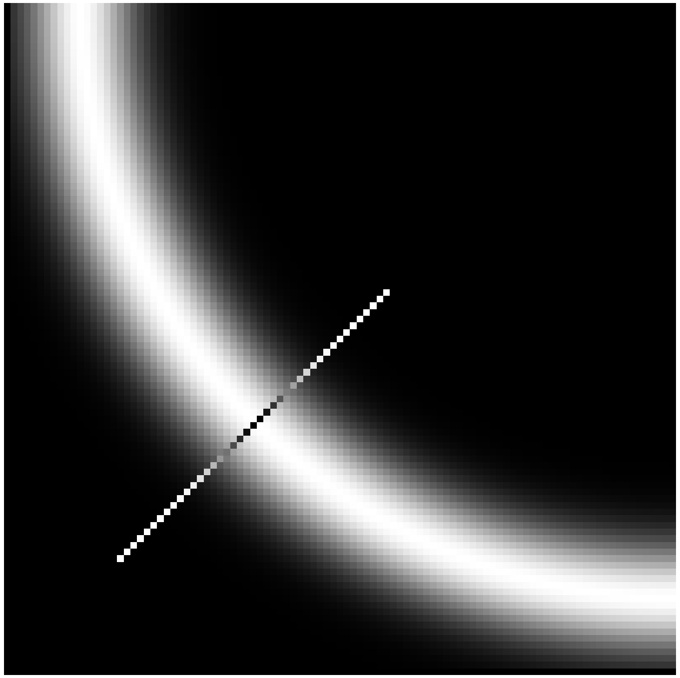
$$\overline{\mathbf{n}} = \frac{\nabla(G * I)}{\left|\nabla(G * I)\right|}$$

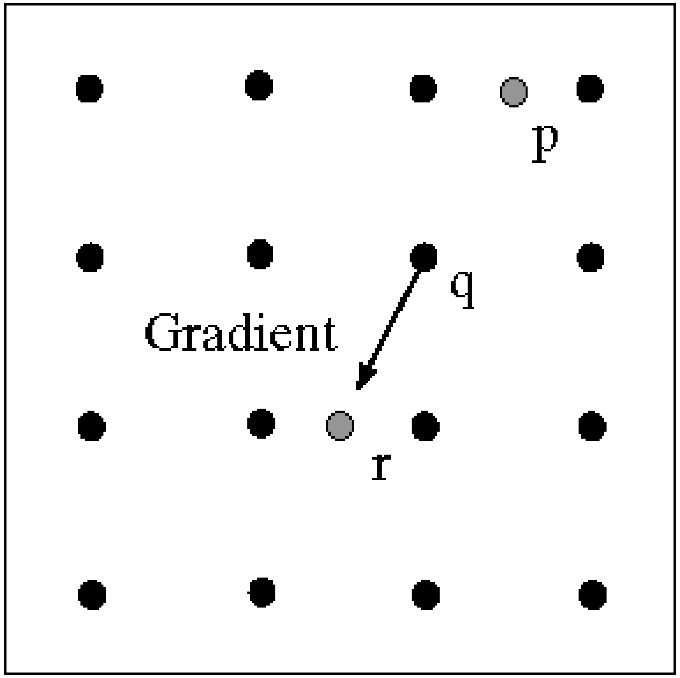
- Compute edge magnitudes $\left|\nabla(G * I)\right|$
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

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

Non-maximum Suppression

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





original image

magnitude of the gradient

After non-maximum suppression

Canny Edge Operator

original

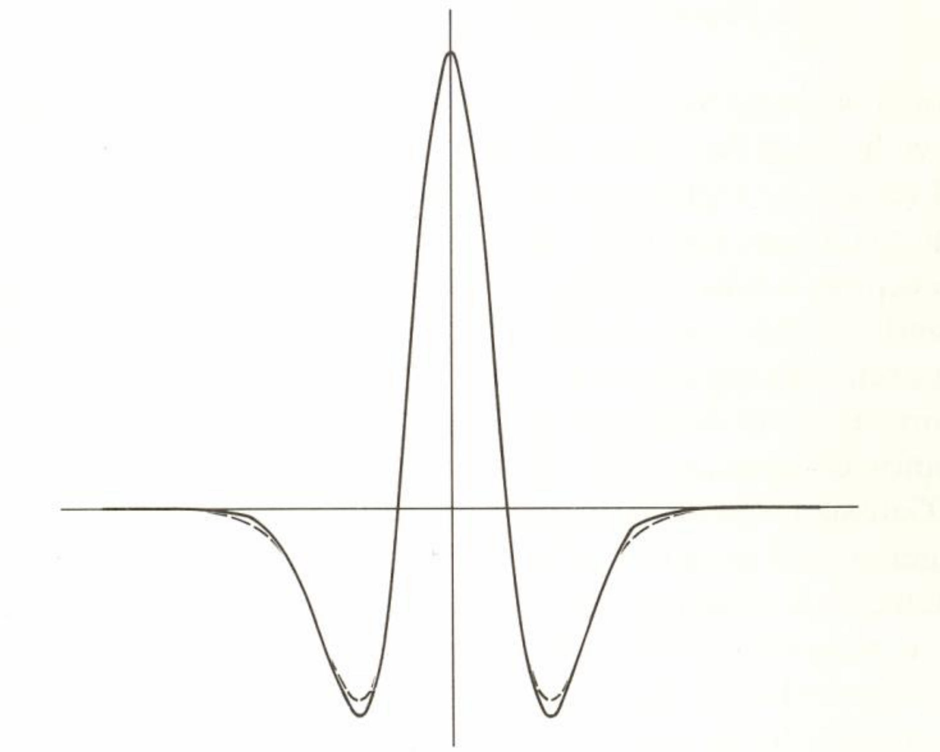
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

Difference of Gaussians (DoG)

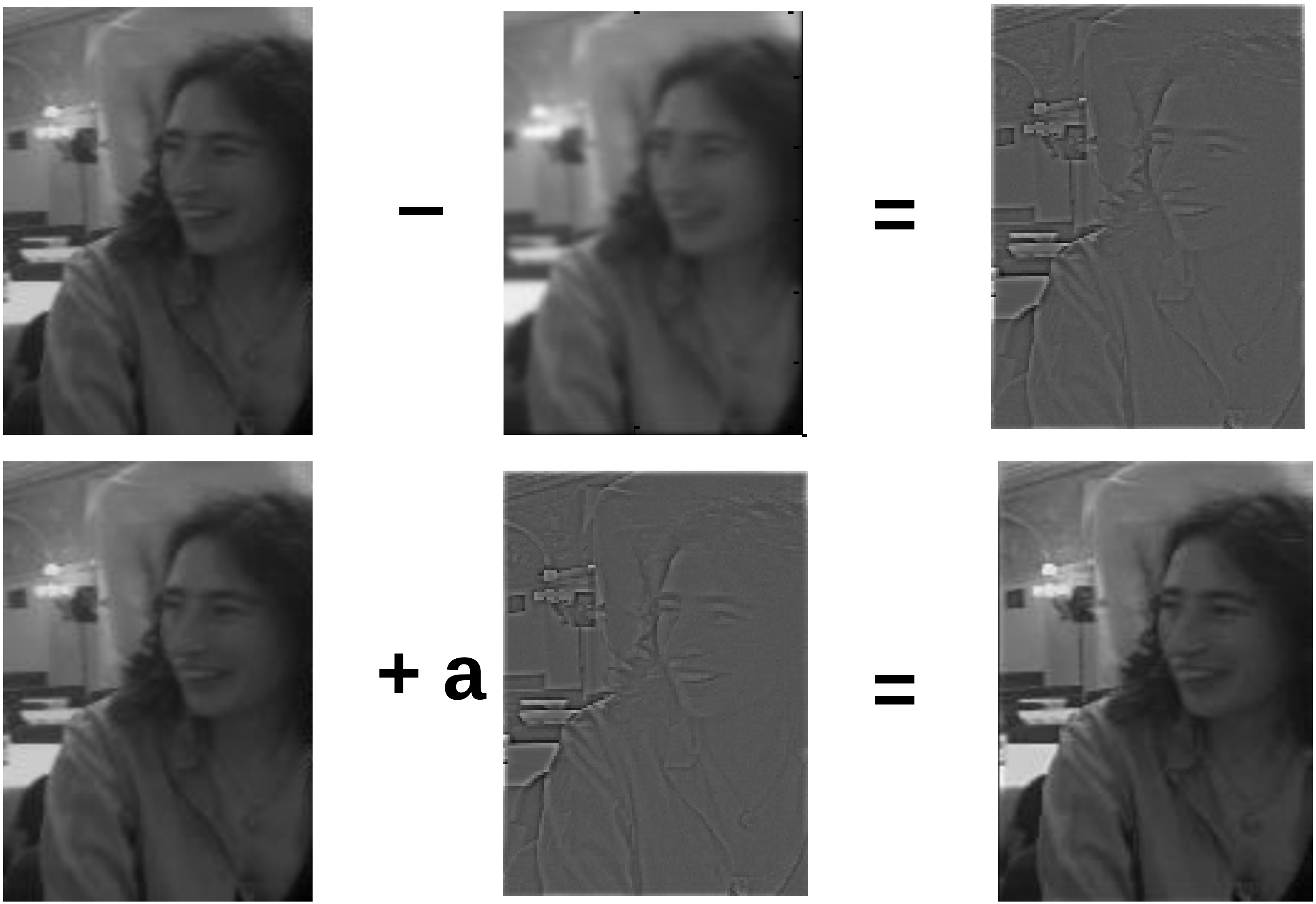
- Laplacian of Gaussian can be approximated by the difference between two different Gaussians



DoG Edge Detection

(a) $\sigma = 1$ (b) $\sigma = 2$ (b)-(a)

Unsharp Masking



Resources

- Gonzalez & Woods – Chapter 4