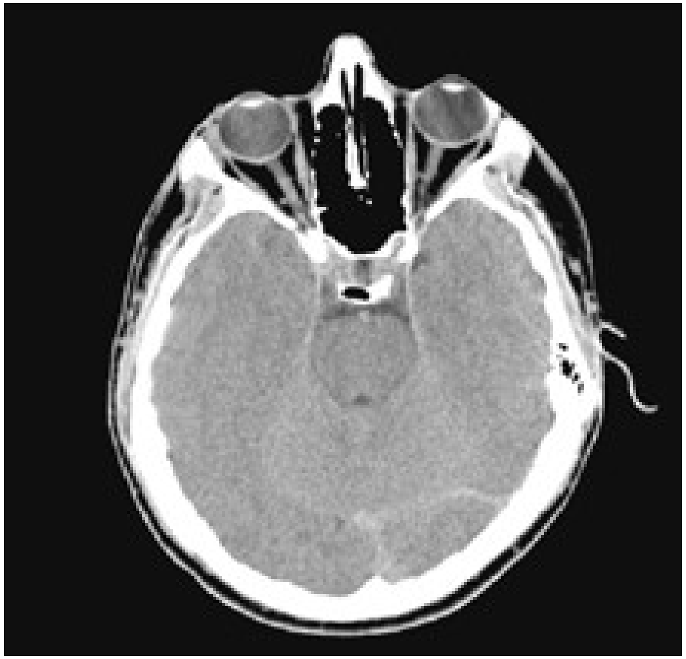
Spatial filtering

- Multidimensional signals
- Spatial convolution
- Smoothing spatial filters
- Examples using smoothing spatial filters
- Smoothing spatial filters
- Spatial filtering of color images
- Sharpening spatial filters
- The first-order derivative for (non-linear) image sharpening – the gradient
- Example of exam task
- The second-order derivatives for image sharpeninig – the Laplacian
- How to avoid negative values of pixels?
- The second-order derivatives for image sharpeninig – the Laplacian
- Sharpening using smoothing filter

Multidimensional signals

- Images
  - Grey-scale images; 2-D: *f(x, y)* depend on several variables such as spatial coordinates *(x, y)*

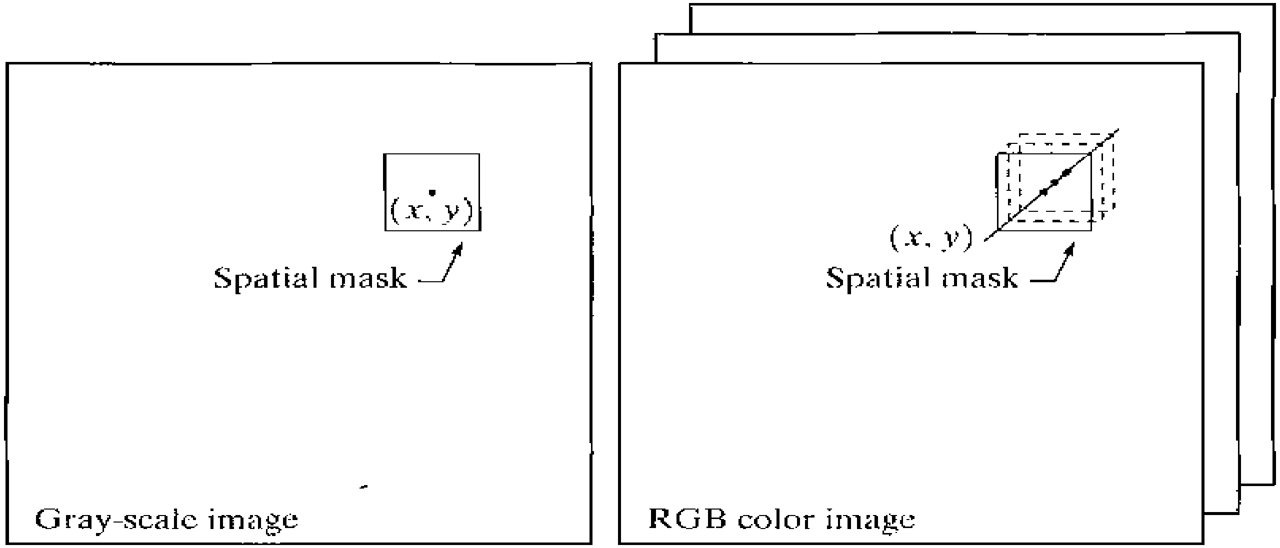


$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Multidimensional signals

- Color images
  - Three channels; red, green, blue; 3 X 2-D: $\{r(x, y), g(x, y), b(x,y)\}$

$$\mathbf{c}(x,y) = \begin{bmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{bmatrix} = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

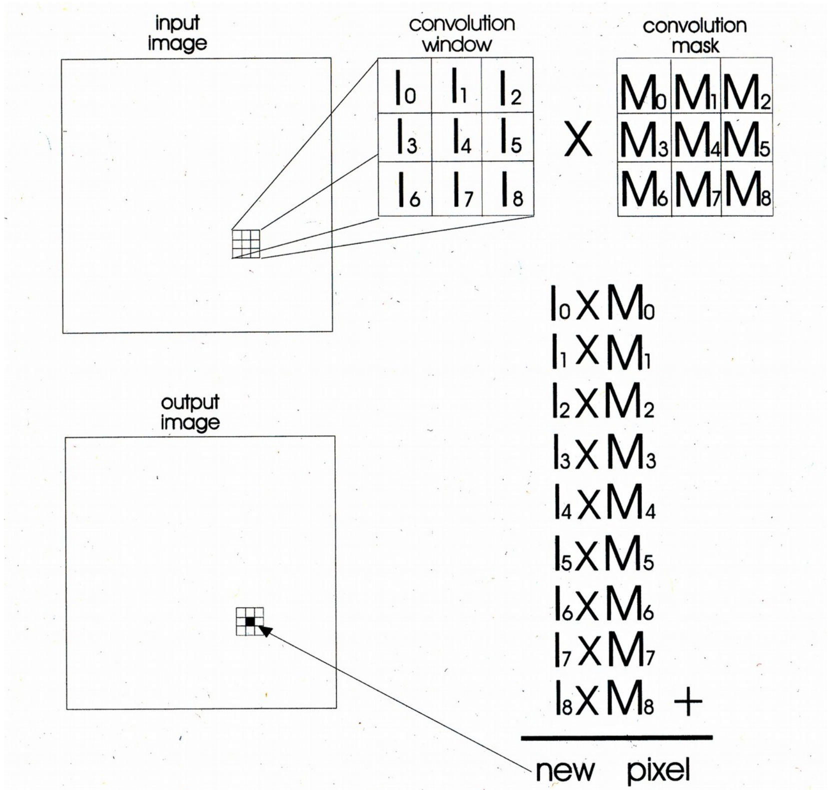


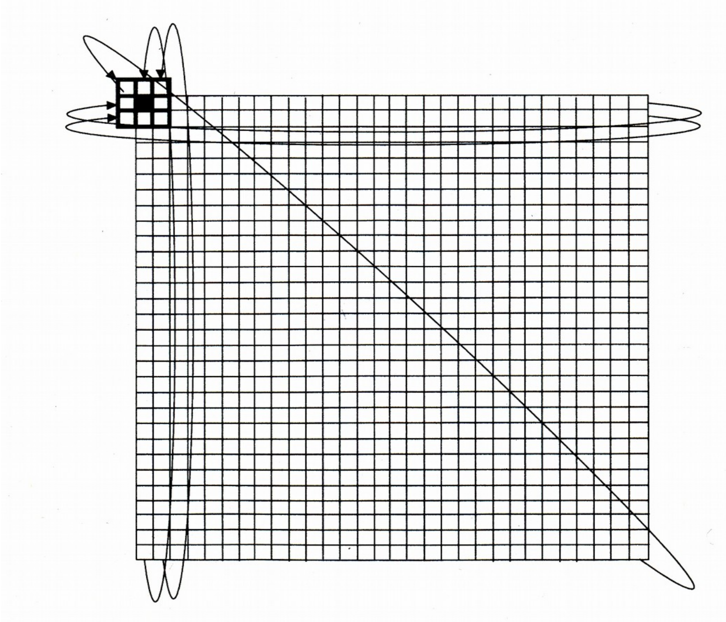
Spatial convolution

- Convolution (convolution kernel, impulse response, spatial mask, template)

$g(x, y) = w(s,t) \ X \ f(x,y)$

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





Smoothing spatial filters

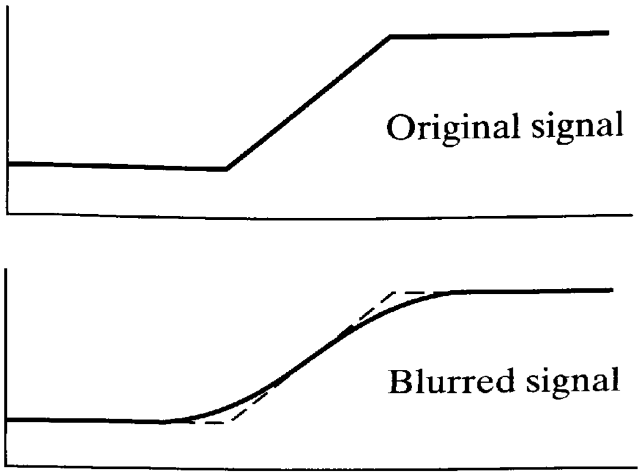
- Smoothing (blurring)
  - Rearranging intensities in image with the aim to smooth sharp peaks
  - Filtering using linear low-pass filters, positive coefficients of the mask
  - Smoothing using moving average (a box filter), smoothing using weighted moving average

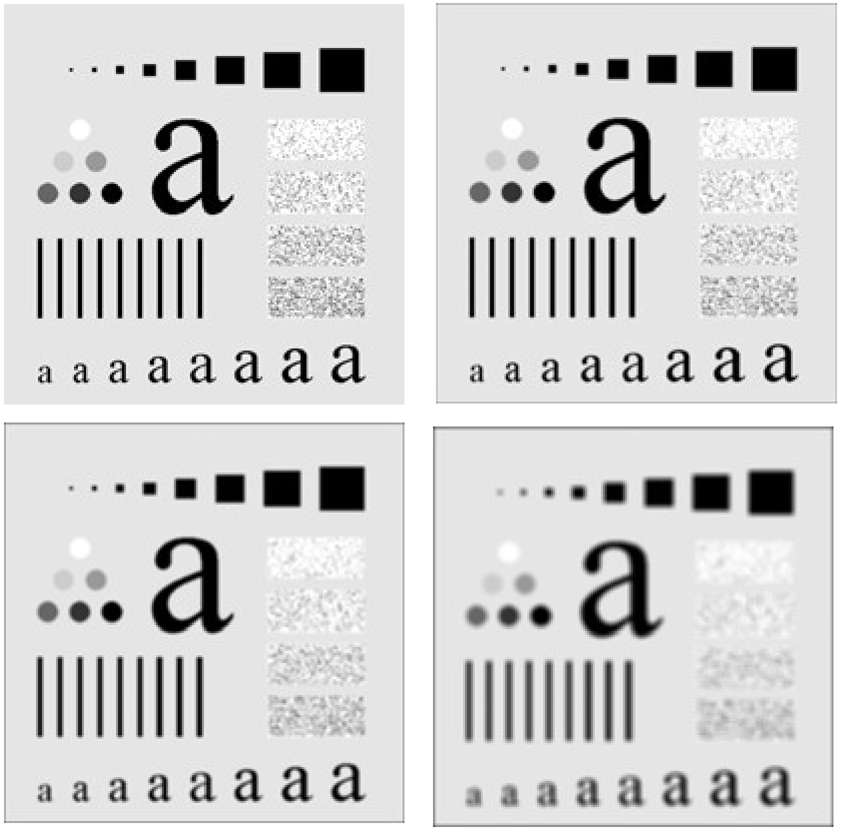
$$\frac{1}{9}\times\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \qquad \frac{1}{16}\times\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

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

Smoothing spatial filters

- Results of smoothing with moving average filters (sizes of spatial masks, M = 3, 5, 9)



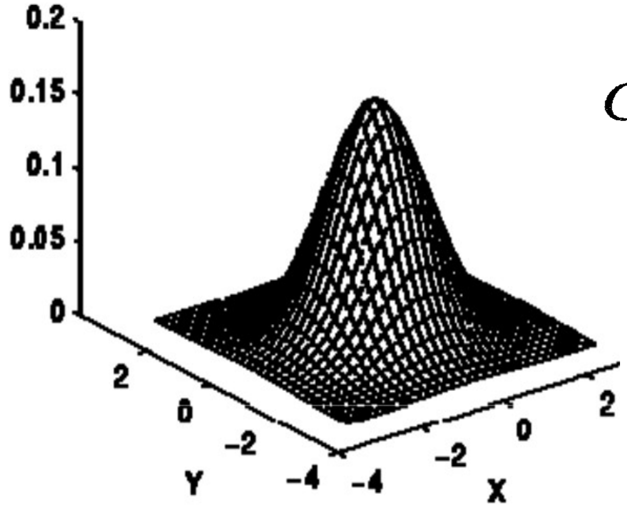


Smoothing spatial filters

- Gaussian filter

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

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$



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

$$\frac{1}{273}$$

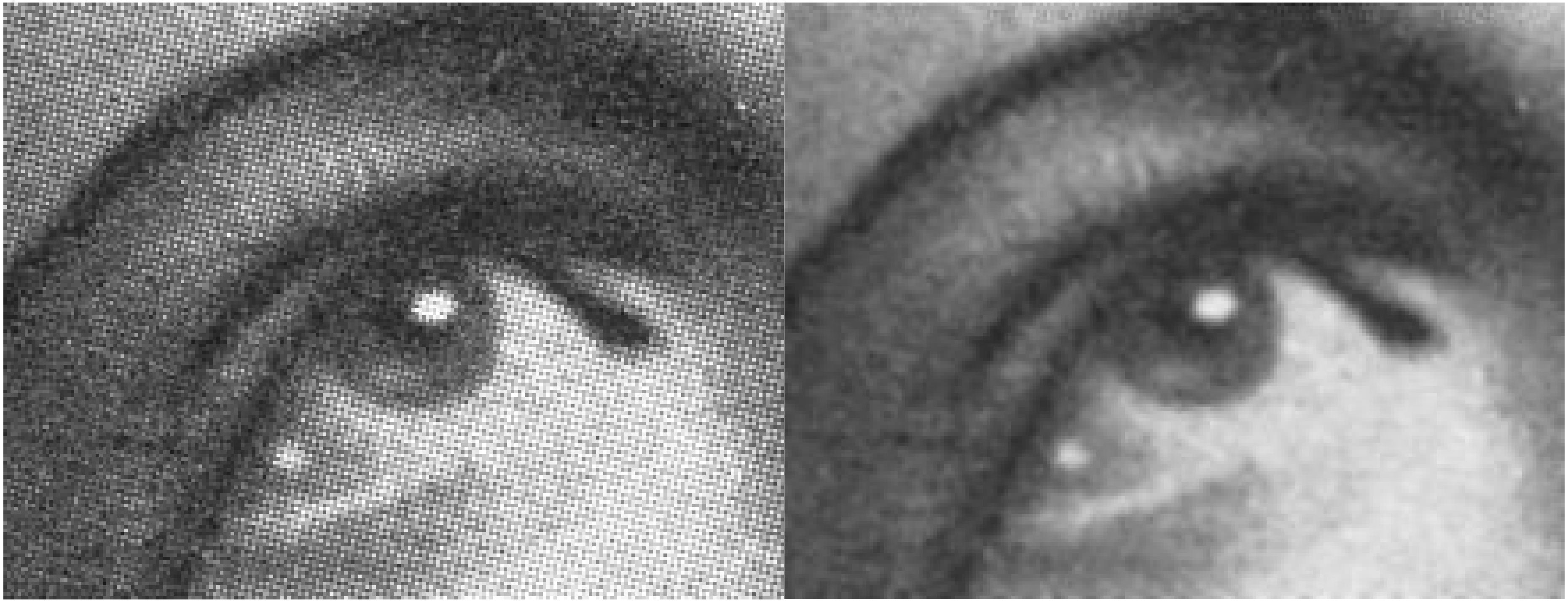
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

Smoothing spatial filters

- Examples

Examples using smoothing spatial filters



Examples using smoothing spatial filters

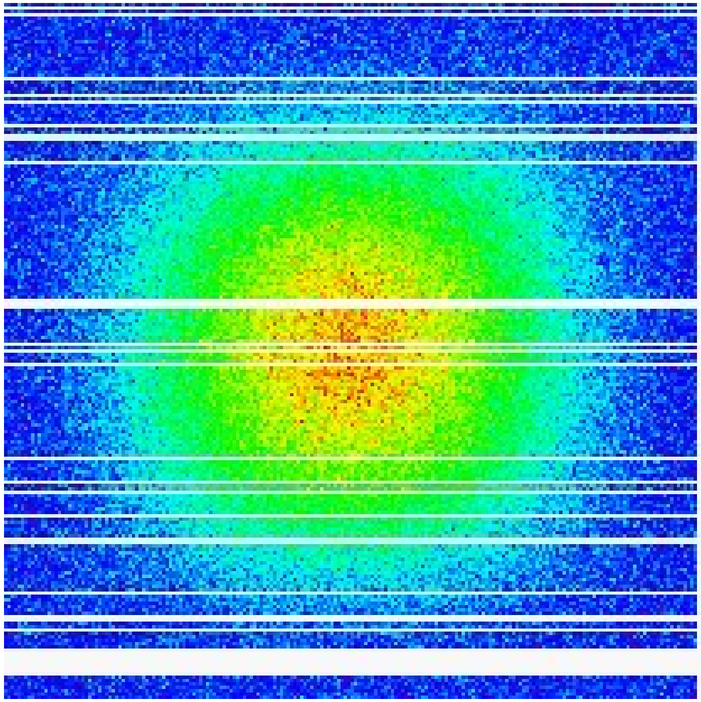
Gaussian Noise

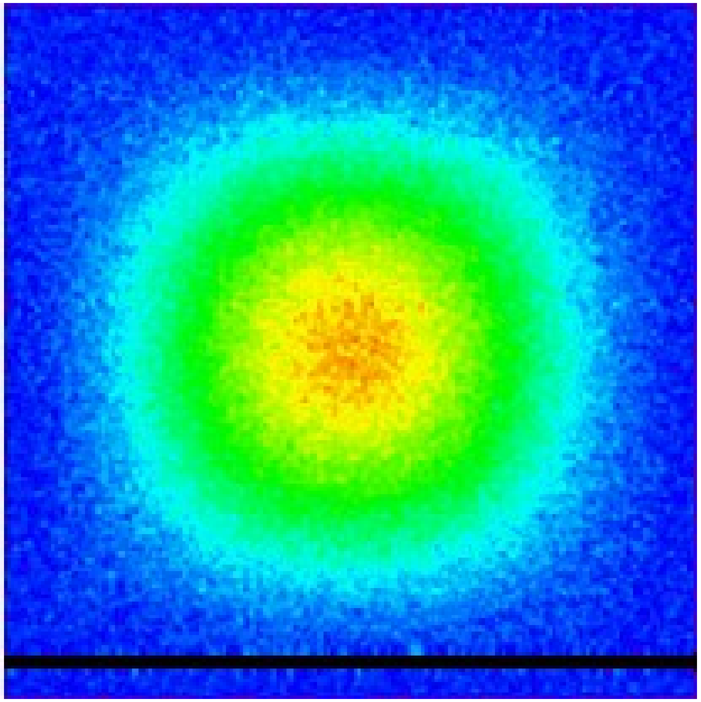
Average Filtered Image

Examples using smoothing spatial filters

Examples using smoothing spatial filters

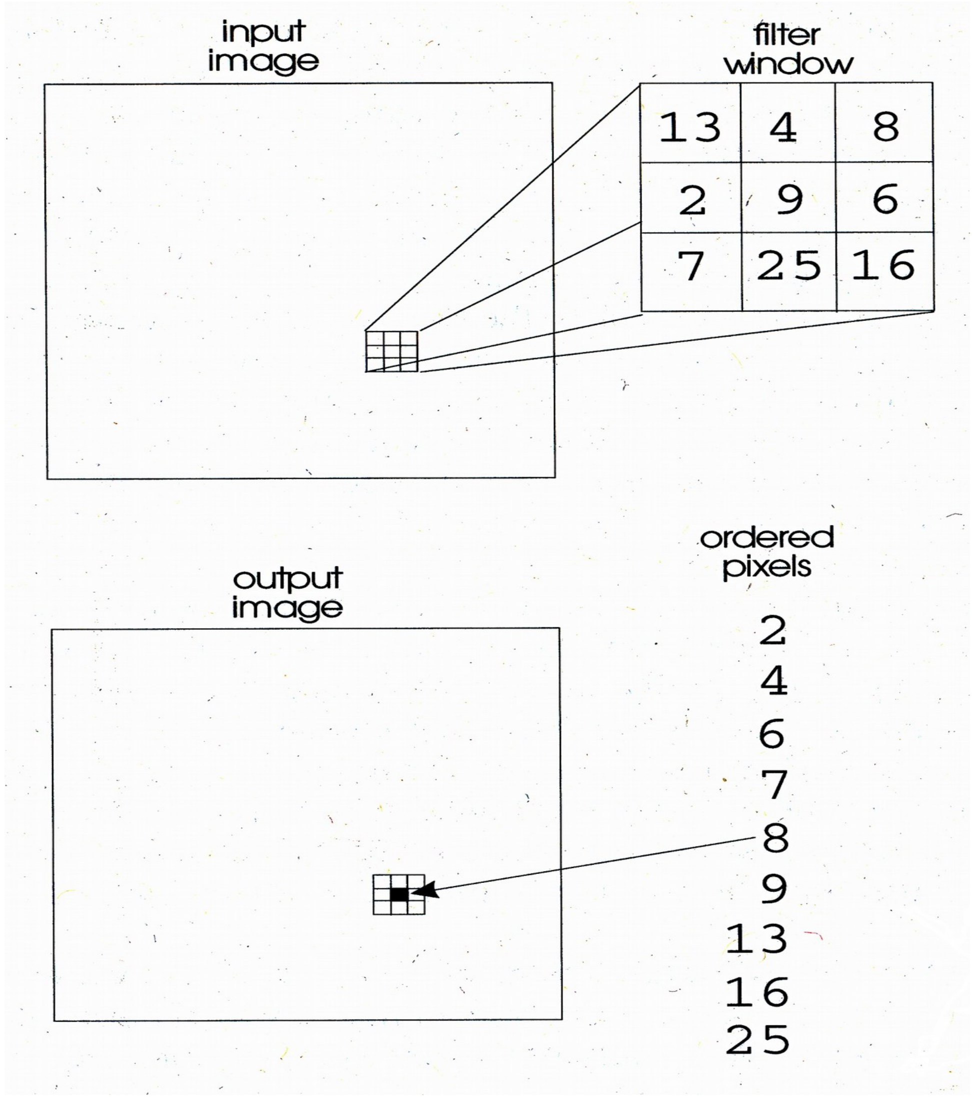
Examples using smoothing spatial filters





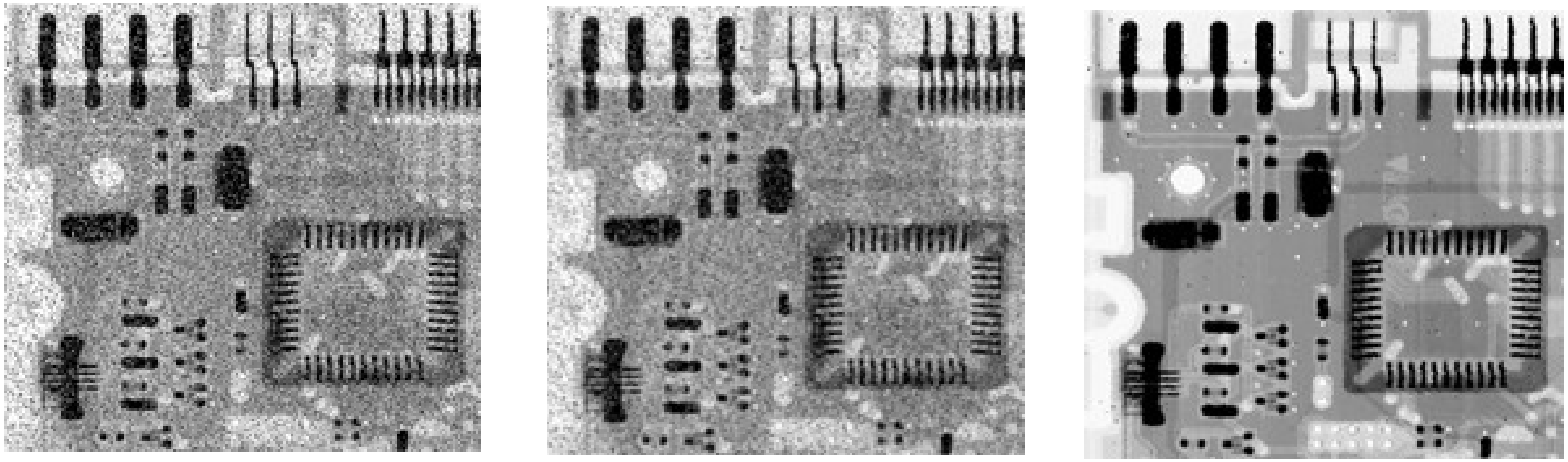
Smoothing spatial filters

- Median filter



Smoothing spatial filters

- Results of smoothing with square averaging filter (size of spatial mask, $M = 3$) and with 3 X 3 median filter



[Gonzales, Woods]

Spatial filtering of color images

- The same operation is performed in each channel
  - Example, moving average filter

  c(*x*,*y*) – *a*n arbitrary vector in RGB color space

  S_{xy} – the set of coordinates defining a neighborhood centered at (*x*, *y*) in an RGB color image

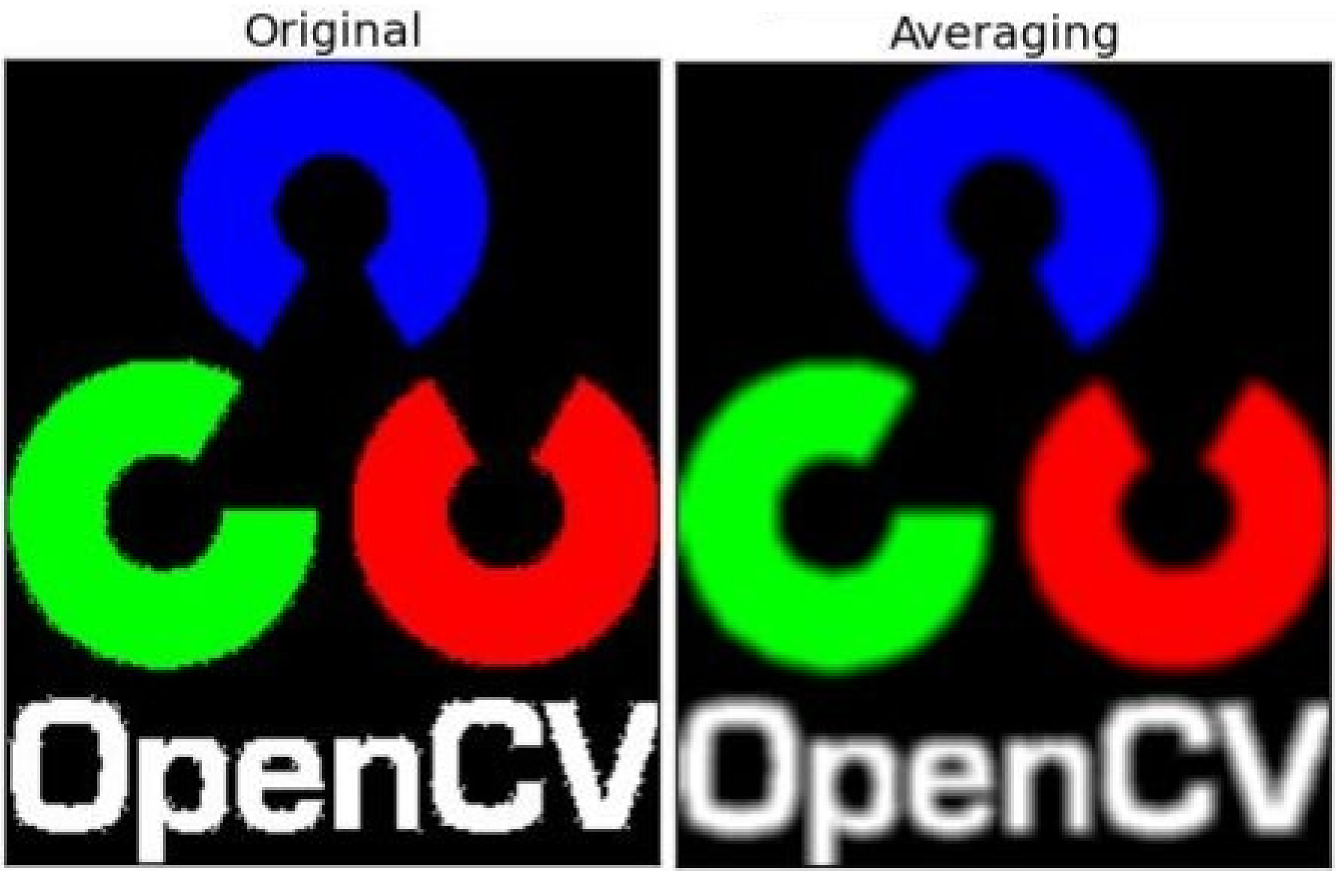
$$\mathbf{c}(x,y) = \begin{bmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{bmatrix} = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

$$\bar{\mathbf{c}}(x,y) = \frac{1}{K} \sum_{(x,y)\in S_{xy}} \mathbf{c}(x,y)$$

$$\bar{\mathbf{c}}(x,y) = \begin{bmatrix} \frac{1}{K} \sum_{(x,y)\in S_{xy}} r(x,y) \\ \frac{1}{K} \sum_{(x,y)\in S_{xy}} g(x,y) \\ \frac{1}{K} \sum_{(x,y)\in S_{xy}} b(x,y) \end{bmatrix}$$

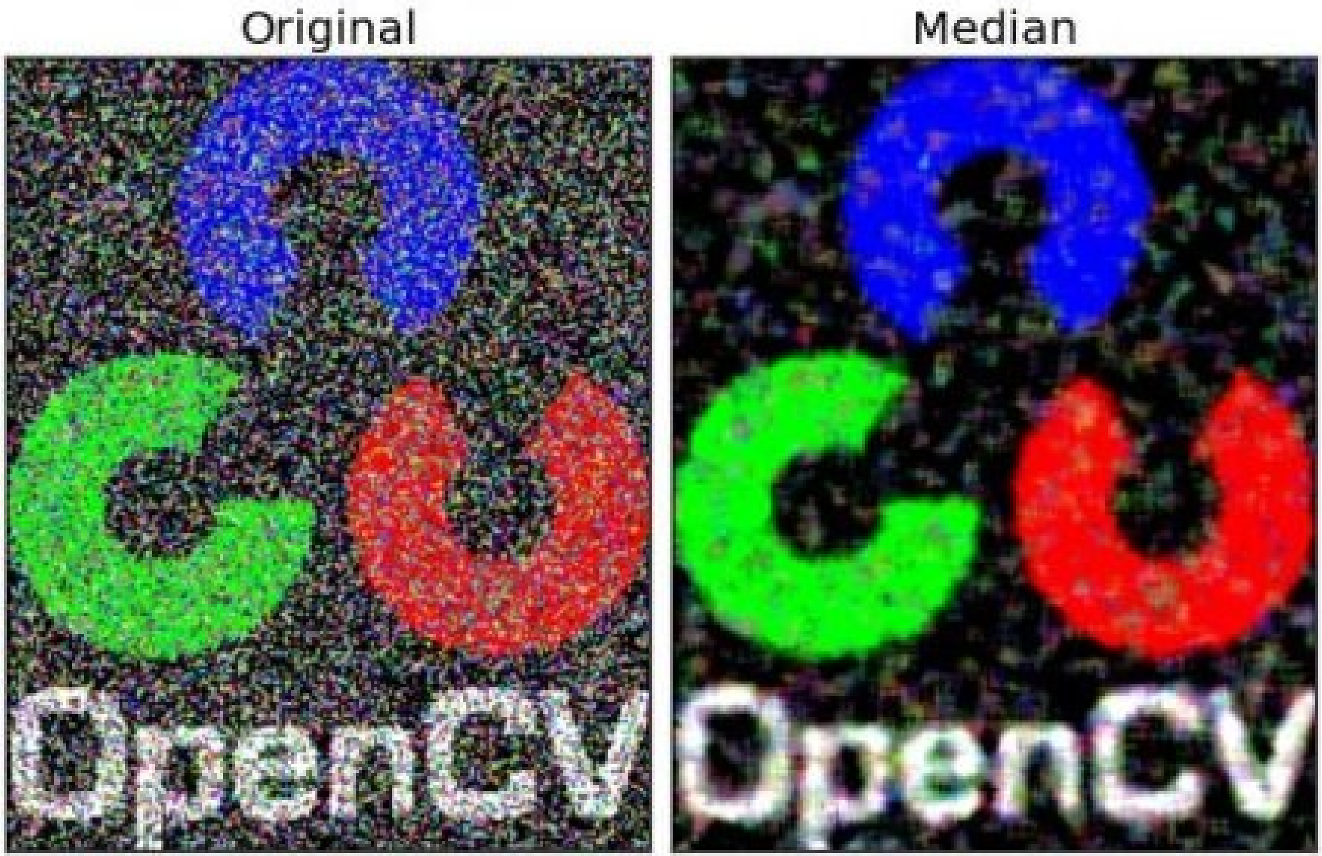
Spatial filtering of color images

- Moving average filter



Spatial filtering of color images

- Example of using median filter



Sharpening spatial filters

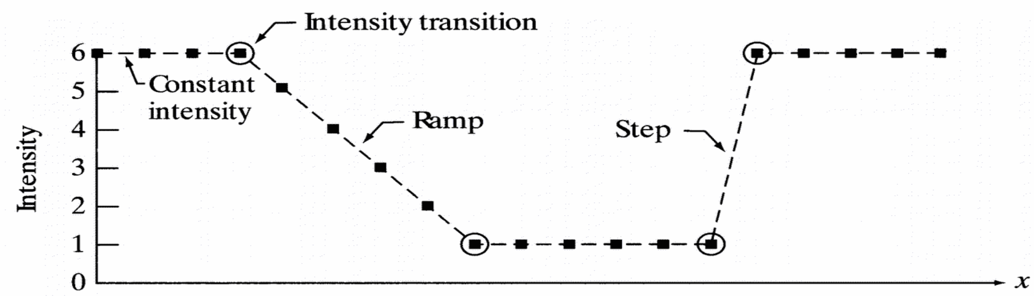
- Sharpening
  - Rearranging intensities in image with the aim to rise differences in intensities of the neighboring pixels to emphasize tiny details
  - Filtering using high-pass filters, first- or second-order derivative

Sharpening spatial filters

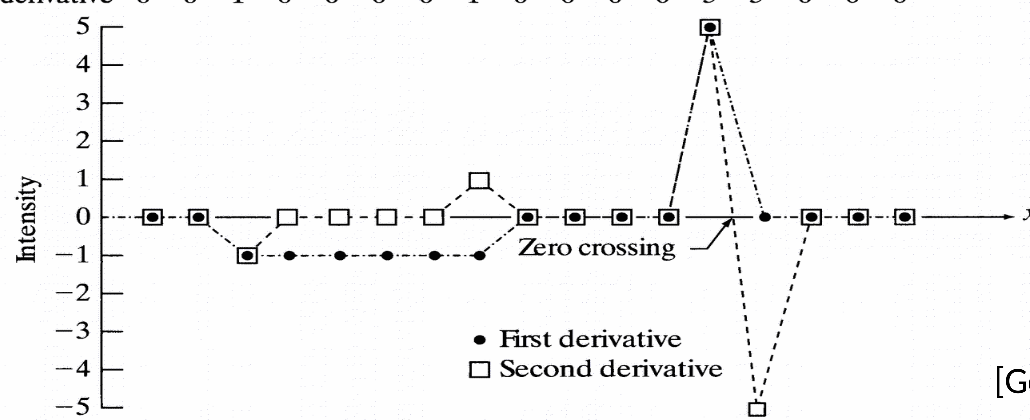
- First- and second-order derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	



[Gonzales, Woods]

The first-order derivatives for (non-linear) image sharpening – the gradient

- Empasizing contours

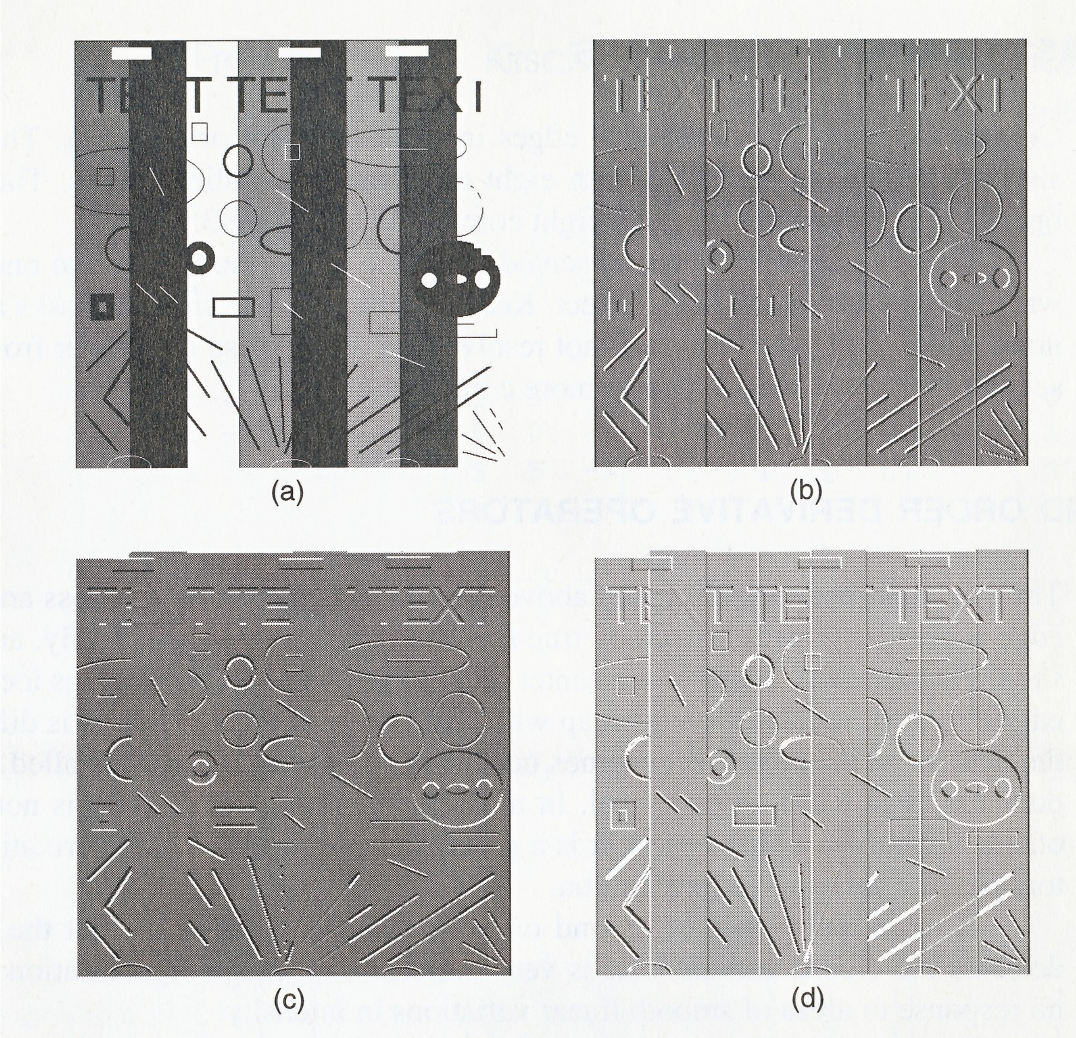
  Prewitt operators

- For raws

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

- For columns

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



(a) (b) (c) (d)

The first-order derivatives for (non-linear) image sharpening – the gradient

- Roberts cross gradient operators, Sobel operators

−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

The first-order derivatives for (non-linear) image sharpening – the gradient

- Magnitude of the gradient, Roberts cross gradient operators, Sobel operators

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

$$\begin{aligned} \nabla f &= \mathrm{mag}(\nabla \mathbf{f}) \\ &= \left[G_x^2 + G_y^2\right]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \end{aligned}$$

$$\nabla f \approx |G_x| + |G_y|$$

−1	0
0	1

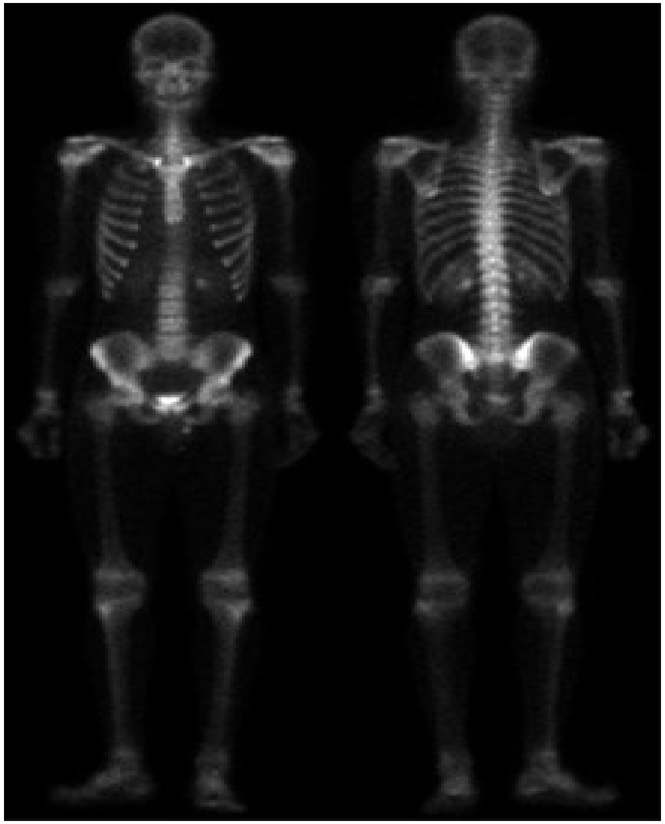
0	−1
1	0

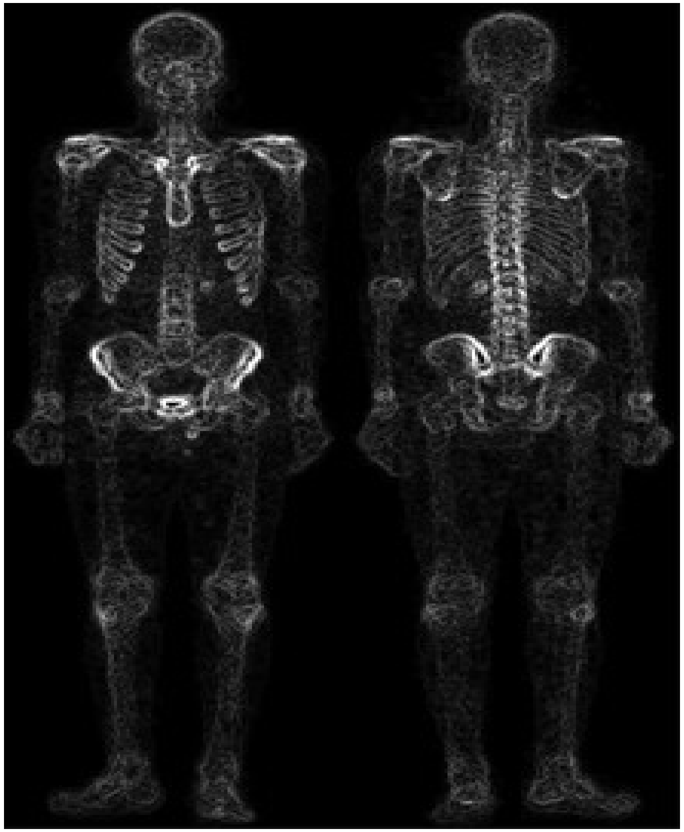
−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

The first-order derivatives for (nonlinear) image sharpening – the gradient

- Sobel gradient



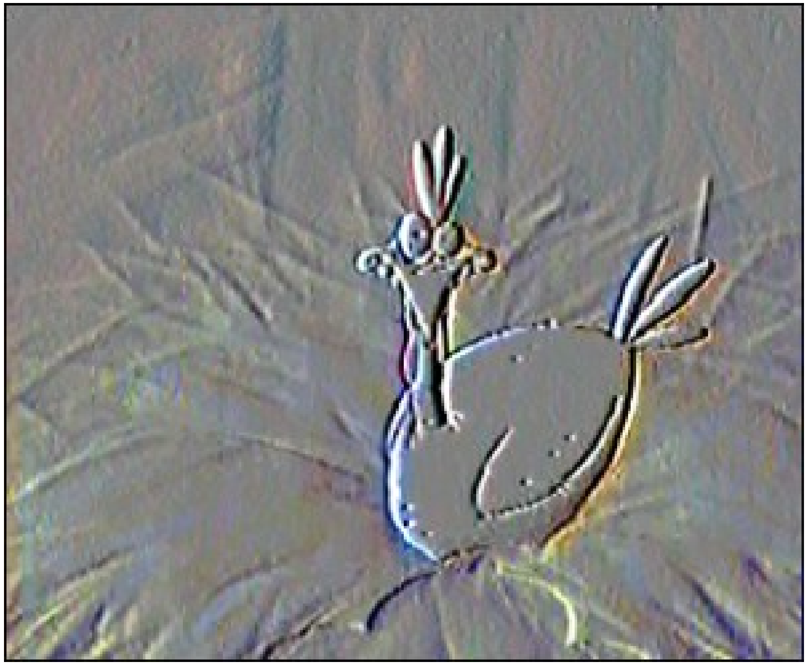


[Gonzales, Woods]

The first-order derivatives for (non-linear) image sharpening – the gradient

- Emphasizing contours

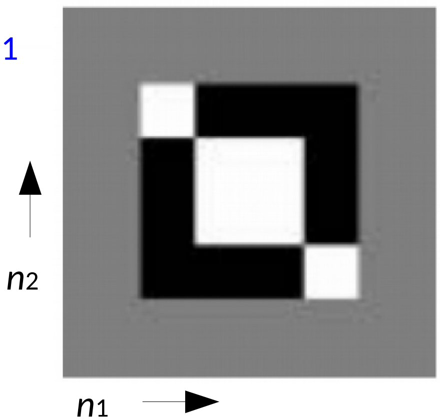
Sobel operators



Example of exam task

Consider the 48 X 48 gray-scale input image in Figure 1, with values coded from -1 for black to +1 for white, as shown. This image contains three values: -1 in the darkest regions, 0 around the border, and 1 in the three light squares. Identify which of the output images in Figure 2 (A, or B, or C, or D ?) results from applying the impulse response $h_1(n_1, n_2)$ of filter H_1 in the sense of convolution to the original input image, and which of the output images in Figure 2 (A, or B, or C, or D ?) results from applying the impulse response $h_2(n_1, n_2)$ of filter H_2 in the sense of convolution to the original input image. Justify both of your answers. The $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$ are defined as:

Figure 1

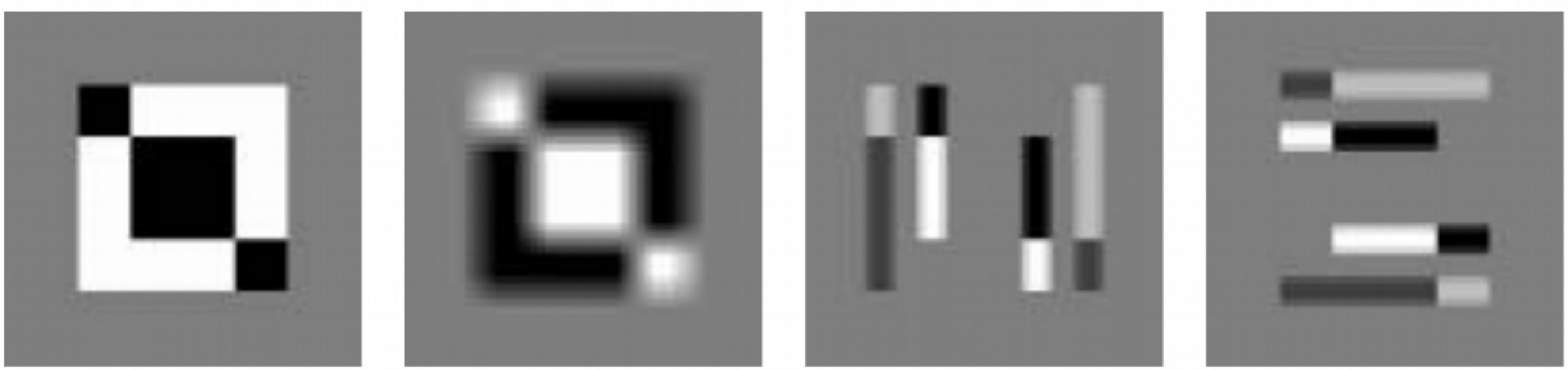


$$h_1[n_1, n_2] = \begin{bmatrix} 0 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{6} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{6} & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \uparrow n_2 \qquad n_1 \rightarrow$$

$$h_2[n_1, n_2] = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix} \uparrow n_2 \qquad n_1 \rightarrow$$

(A) (B) (C) (D)

Figure 2

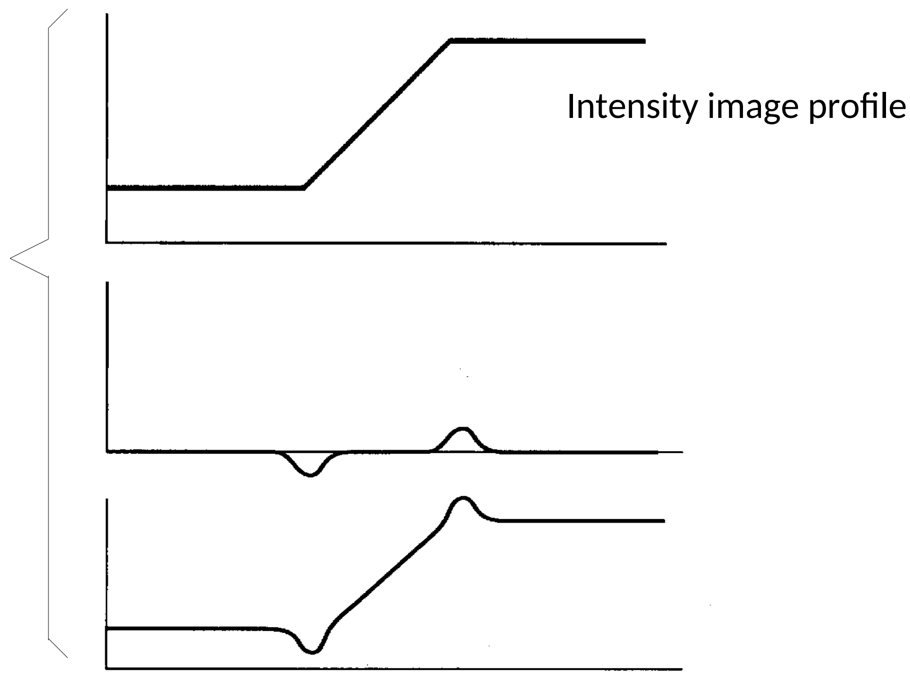


The second-order derivatives for image sharpening – the Laplacian

- Sharpening
  - Rearranging intensities in image with the aim to rise differences in intensities of the neighboring pixels to emphasize tiny details
  - Filtering using high-pass filters, second order derivative, central coefficients positive and neighboring coefficients negative (or vice versa), sum of the coefficients equals zero

(*x* axis)

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$



The second-order derivatives for image sharpening – the Laplacian

- The Laplacian operator
  (2D second-order derivative)

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

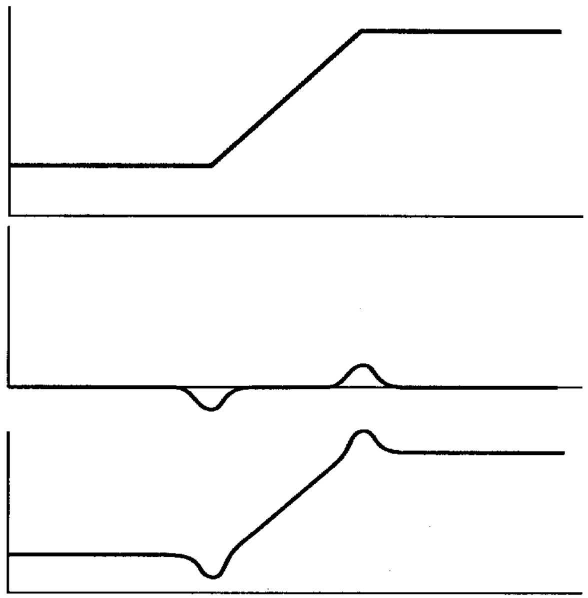
$$\frac{\partial^2 f}{\partial^2 x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial^2 y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

The second-order derivatives for image sharpening – the Laplacian

- The Laplacian operator

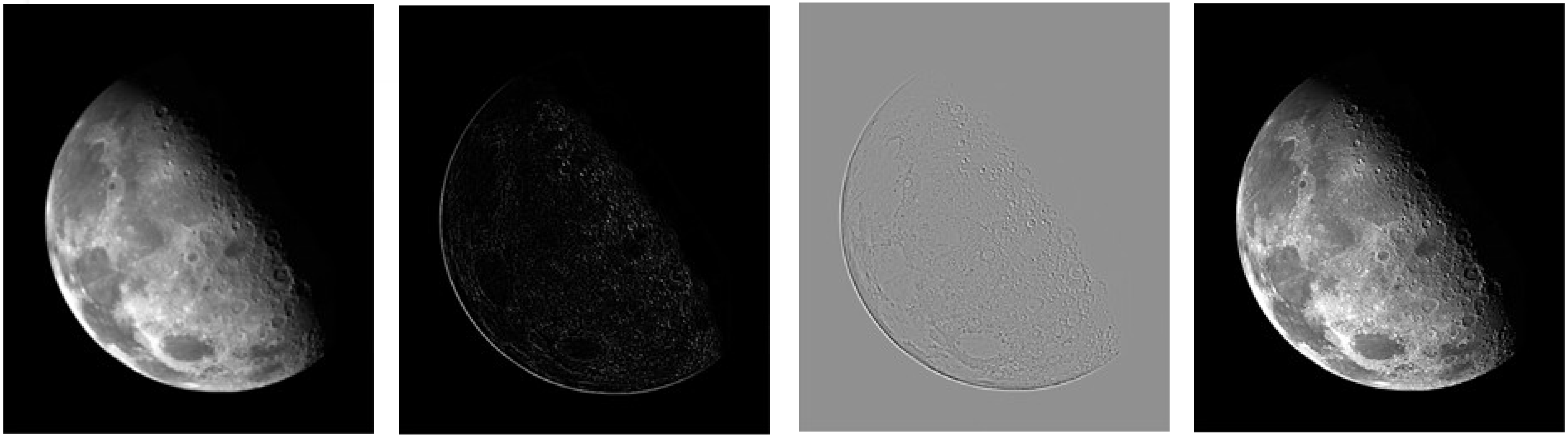


0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x,y) + \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

The second-order derivatives for image sharpening – the Laplacian

- Image sharpening using the Laplacian, original image, Laplacian without scaling, Laplacian with scaling (rise, scale and truncate), sharpened image



1	1	1
1	−8	1
1	1	1

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

[Gonzales, Woods]

How to avoid negative values of pixels?

- How to display images of which values of pixels are negative or above the value of $2^n - 1$?
  (n – number of bits, n = 8)

Rise and truncate

1. Add a constant of $2^n / 2$ to the value of each pixel of an image:

$$Value = Value + 2^n / 2$$

2. Truncate the values of pixels of the image:

$$\text{if } (Value < 0) \quad \text{then } Value = 0,$$
$$\text{if } (Value > 2^n - 1) \quad \text{then } Value = 2^n - 1$$

Move and scale

1. Move the values of pixels of an image, i.e, create an image, f_m, whose minimum value is 0:

$$f_m = f - min(f)$$

2. Scale the values of pixels of the image f_m to fit between 0 and $2^n - 1$:

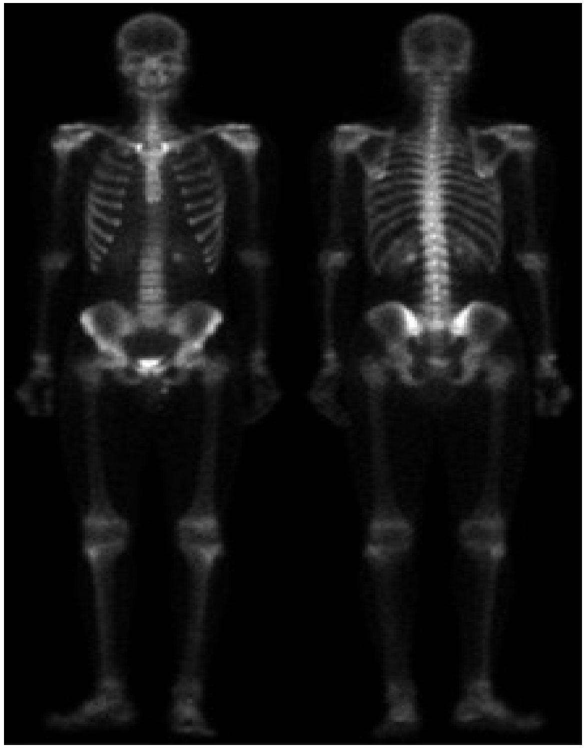
$$f_s = K \,.\, [\, f_m / max(f_m) \,], \quad K = 2^n - 1$$

Rise, scale and truncate

$$f_s = (f + K) / 2, \quad K = 2^n - 1$$

The second-order derivatives for image sharpening – the Laplacian

- Image of whole body bone scan, Laplacian of the image, sharpened image



[Gonzales, Woods]

The second-order derivatives for image sharpening – the Laplacian

- Example using Laplacian

The second-order derivatives for image sharpening – the Laplacian

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x,y) + \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)$$

$$\begin{aligned} g(x,y) &= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] + 4f(x,y) \\ &= 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] \end{aligned}$$

Laplace

0	1	0
1	−4	1
0	1	0

Joint mask

0	−1	0
−1	5	−1
0	−1	0

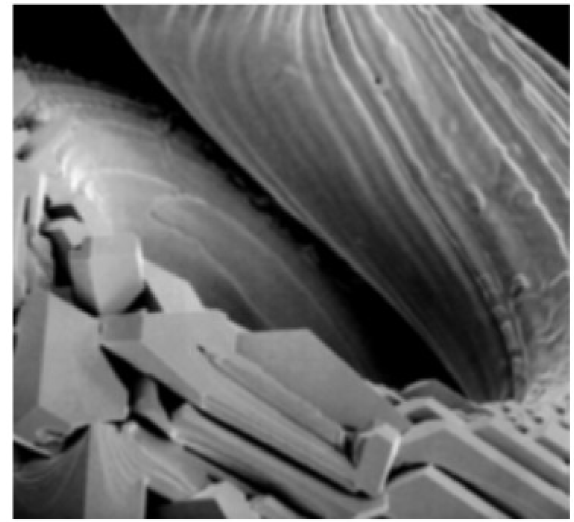
Joint mask with diagonals

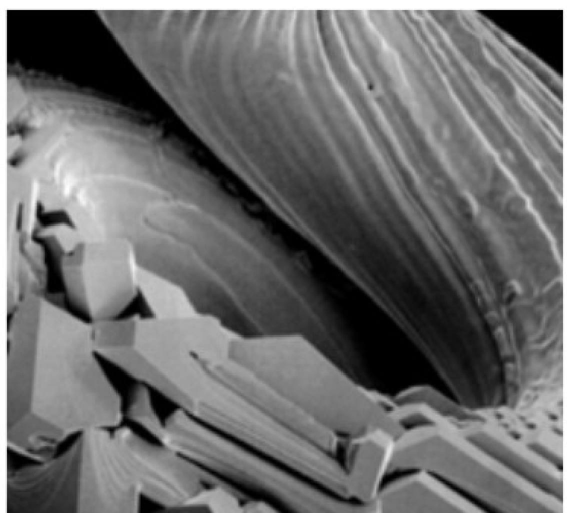
−1	−1	−1
−1	9	−1
−1	−1	−1

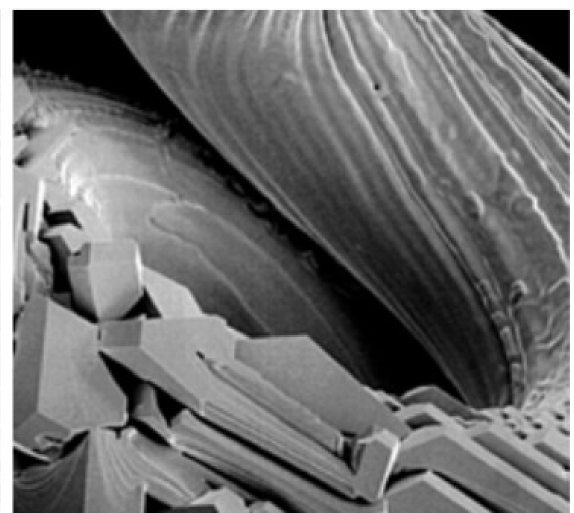
The second-order derivatives for image sharpening – the Laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$







 [Gonzales, Woods]

Sharpening using smoothing filter

1. Blur the original image *f(x,y)*
2. Subtract the blurred image *f*b*(x,y)* from the original (→ the *mask*)

   Unsharp masking:

   *f*s*(x,y)* = *f(x,y)* - *f*b*(x,y)*
3. Add the mask to the original:

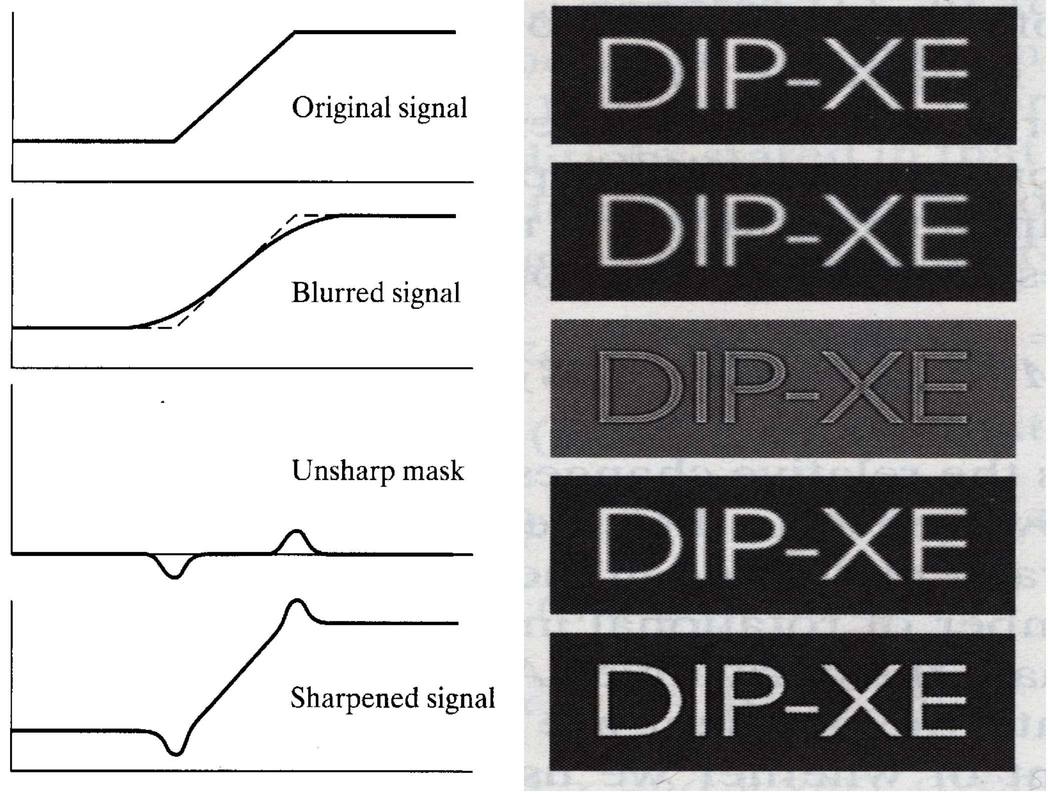
   g(x,y) = *f(x,y)* + *f*s*(x,y)*

Highboost filtering *(A > 1):*

*f*hb*(x,y)* = *A* . *f*s*(x,y)*

3. Add the mask to the original:

   g(x,y) = *f(x,y)* + *f*hb*(x,y)*



Original image, result of blurring with a Gaussian filter, unsharp mask, result of using unsharp masking, result of using highboost filtering

[Gonzales, Woods]