Image filtering

In Fourier domain

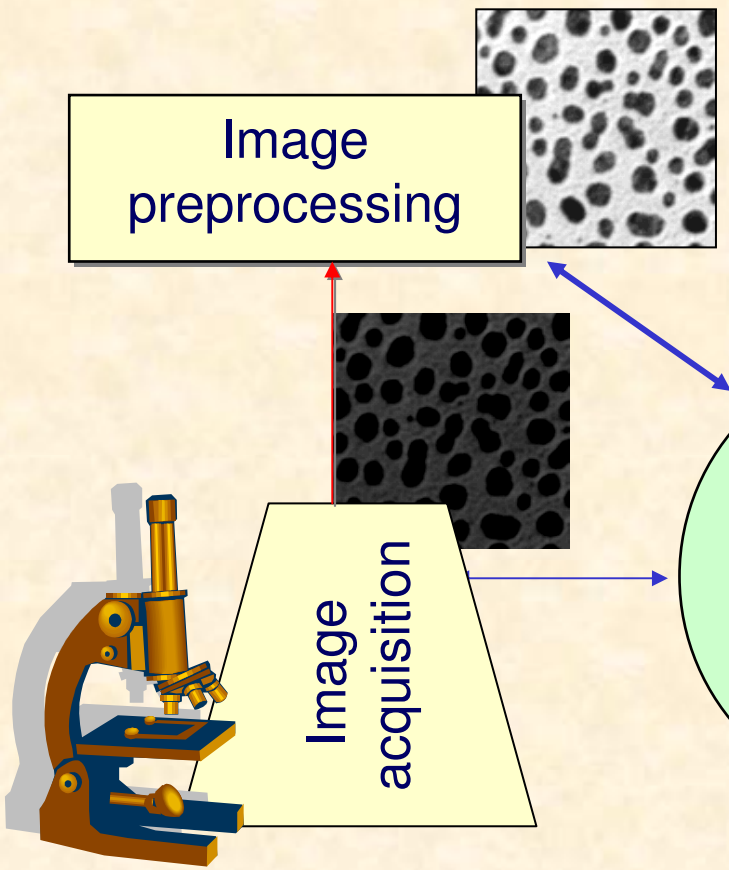
In spatial domain

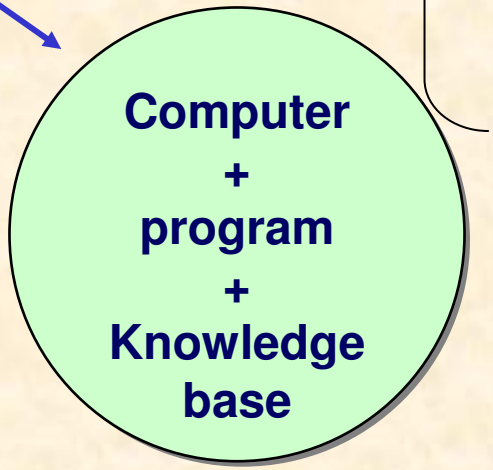
Linear filters

Non-linear filters

P. Strumiłło, M. Strzelecki

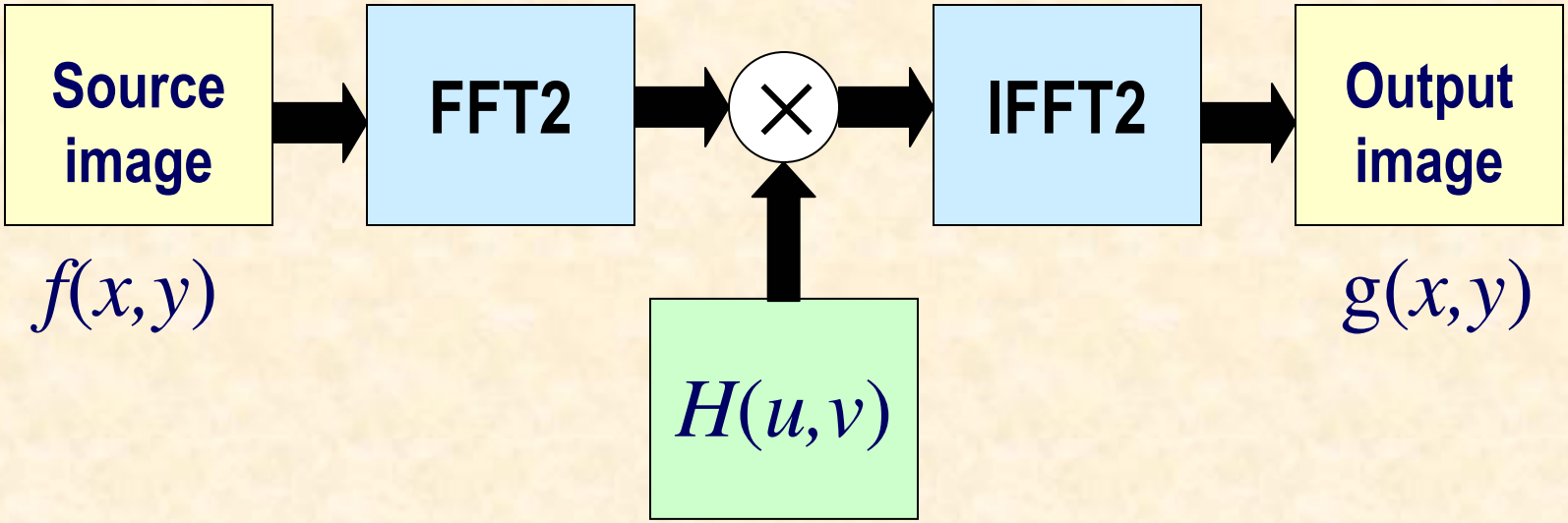
An image processing system





- contrast enhancement
- noise reduction
- image sharpening
- image smoothing
- edge detection
- correction of geometric distortions

Image filtering in spectrum domain

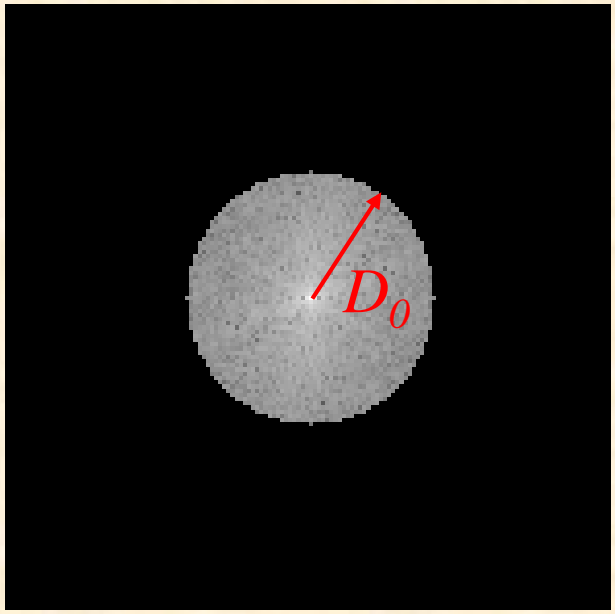


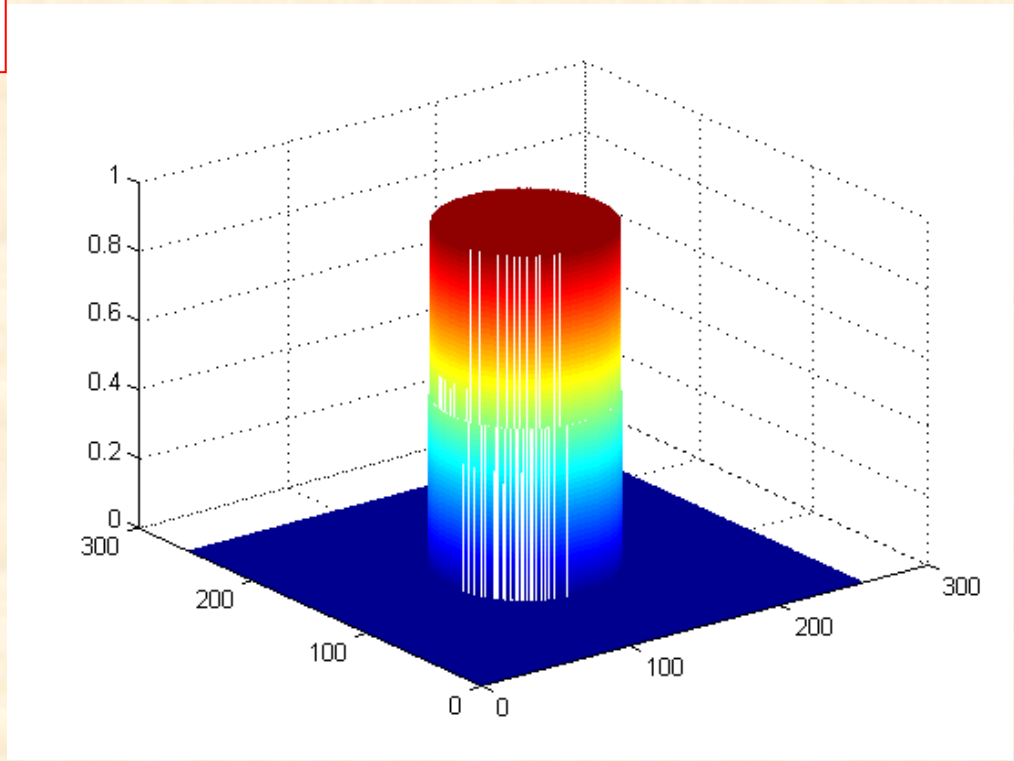
$$g(x,y) = IFFT\{\ H(u,v)\ F\{f(x,y)\}\ \}$$

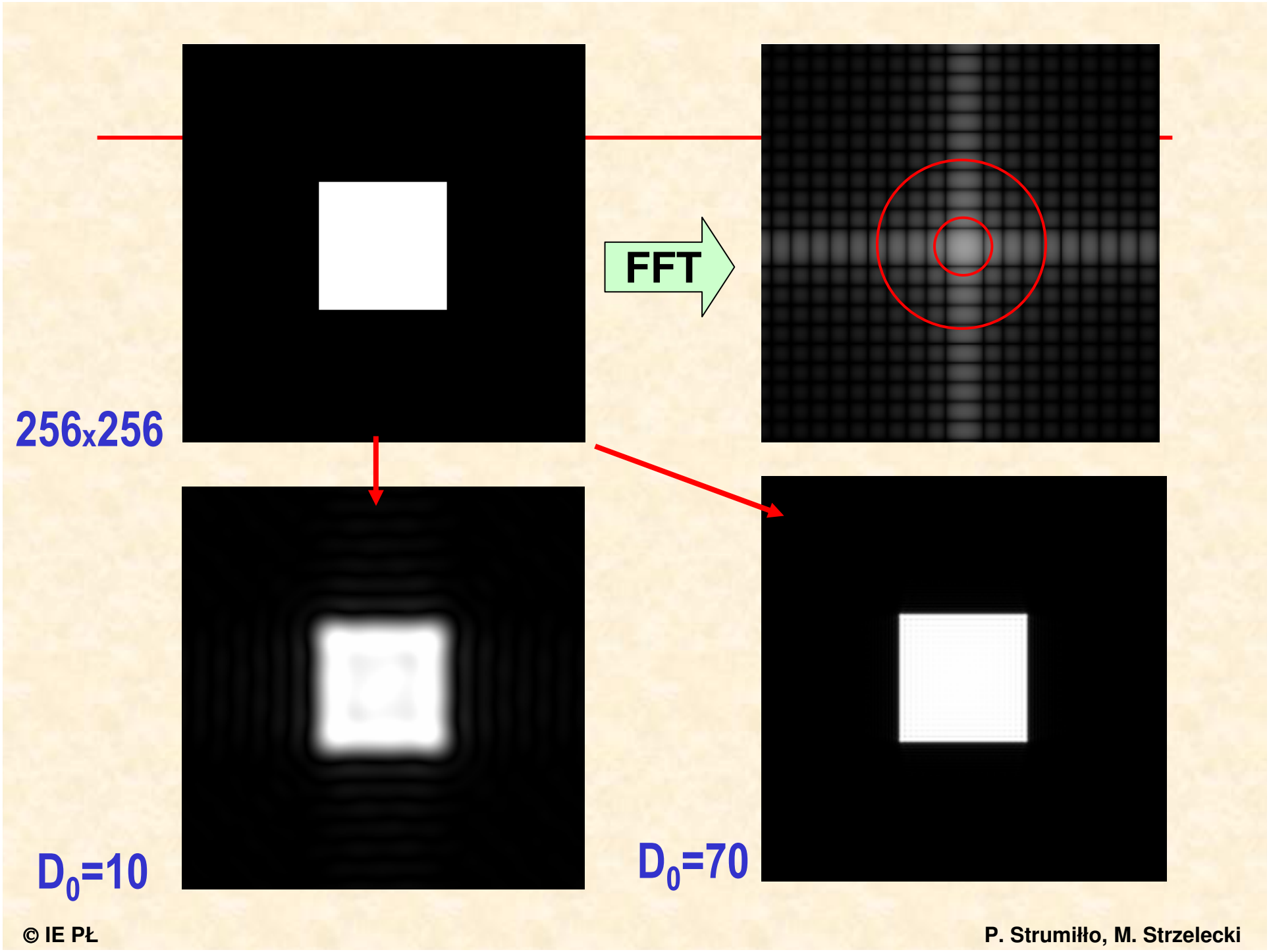
Ideal low-pass filter

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \sqrt{u^2 + v^2}$$







P. Strumiłło, M. Strzelecki

Image after ideal low-pass filtering, D_0=70

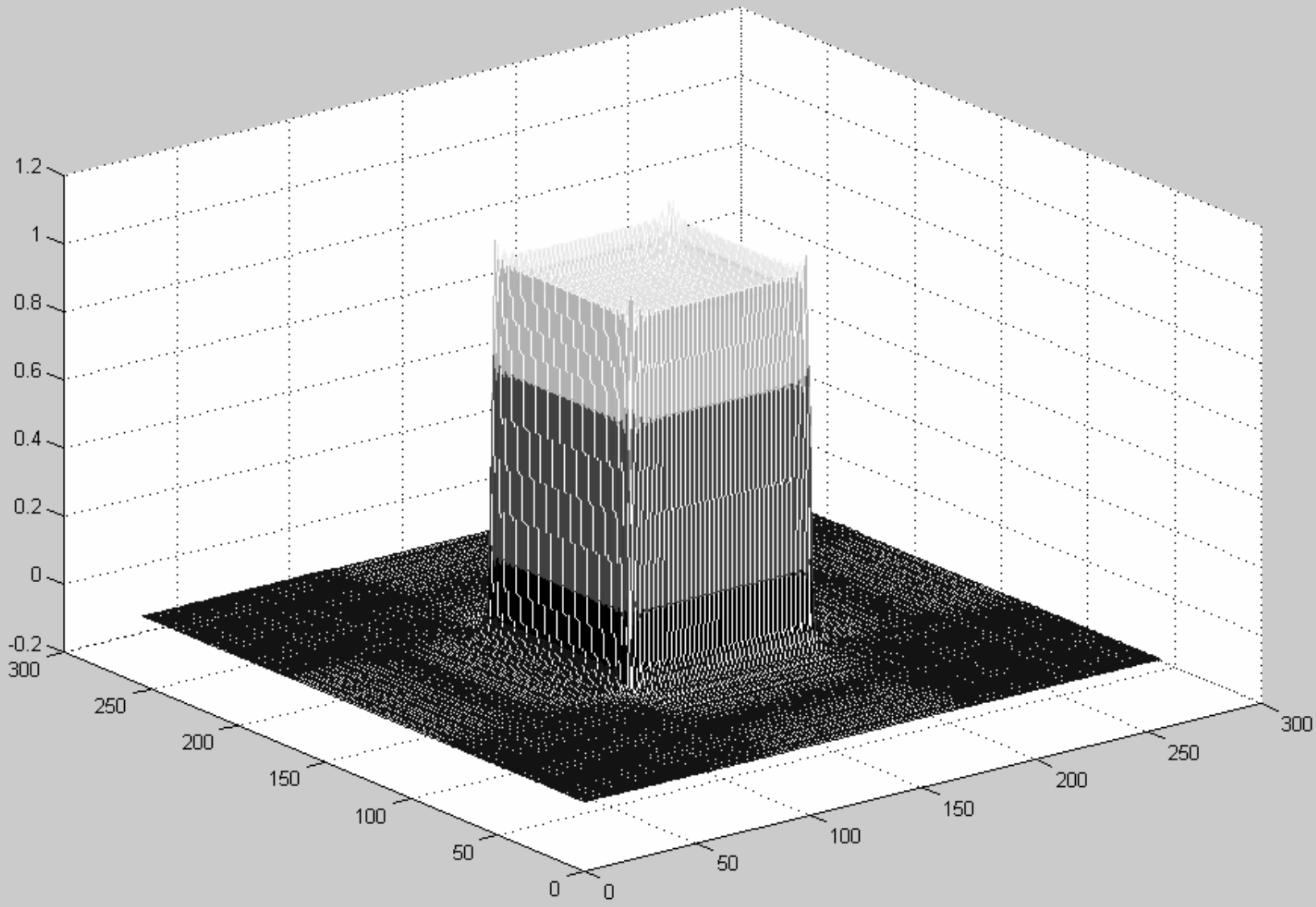
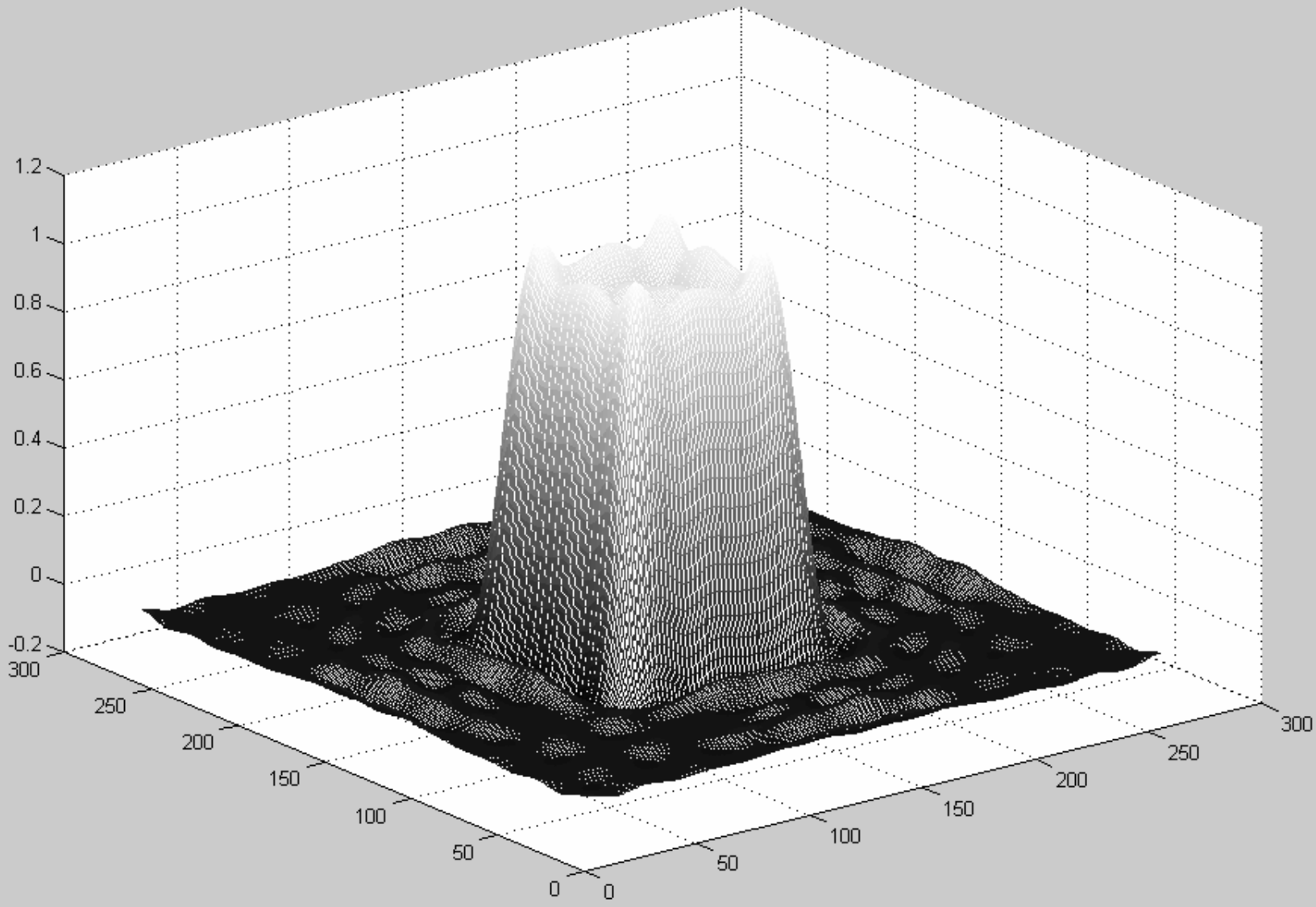


Image after ideal low-pass filtering, D_0=10



Ideal low-pass filter - example

$D_0=10$

$D_0=70$

ringing

$D_0=30$

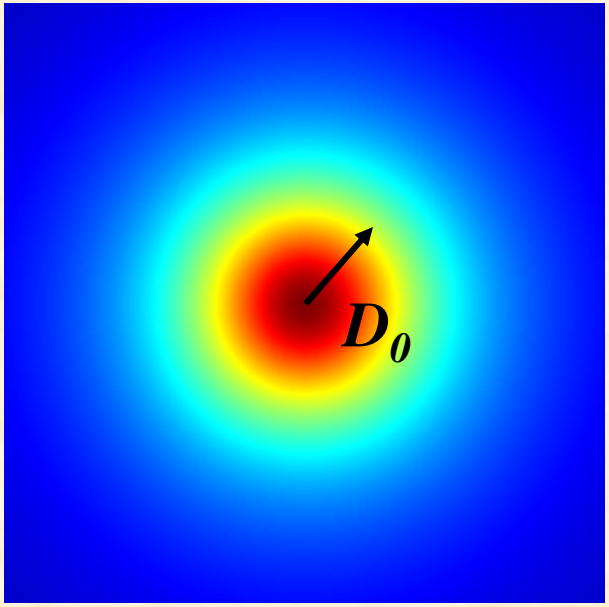
P. Strumiłło, M. Strzelecki

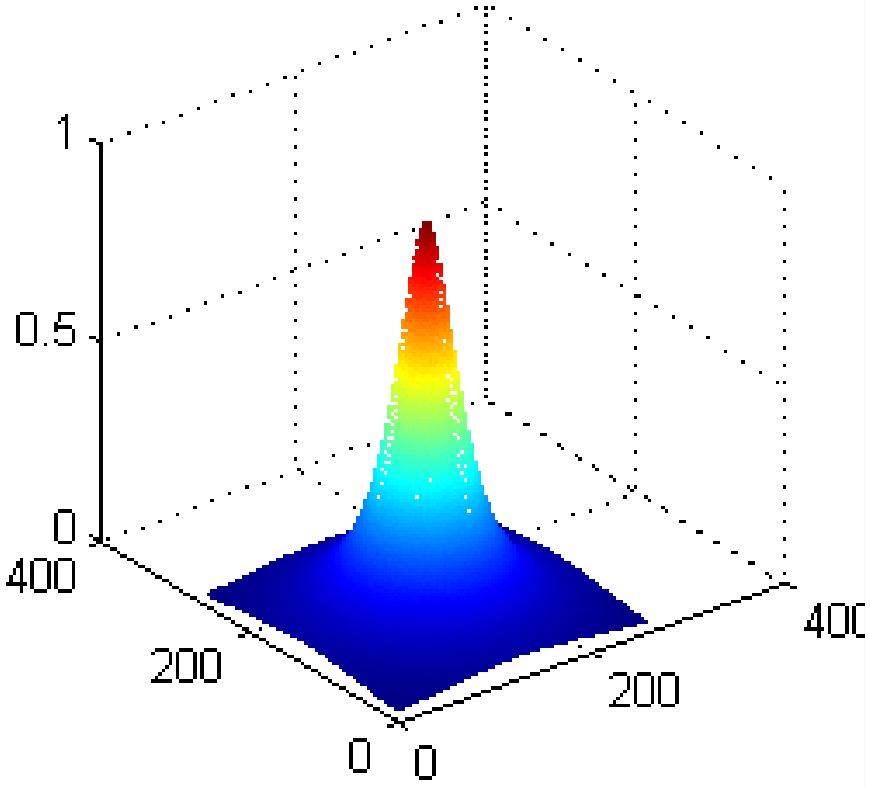
Butterworth filter

$$H(u,v) = \frac{1}{1+(\sqrt{2}-1)[D(u,v)/D_0]^{2n}}$$

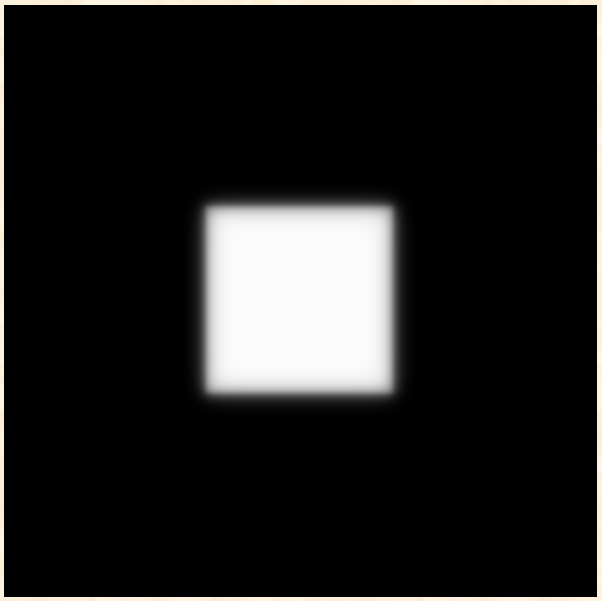
$$D(u,v) = \sqrt{u^2+v^2}, \quad n = 1,2,\ldots$$

n- filter order



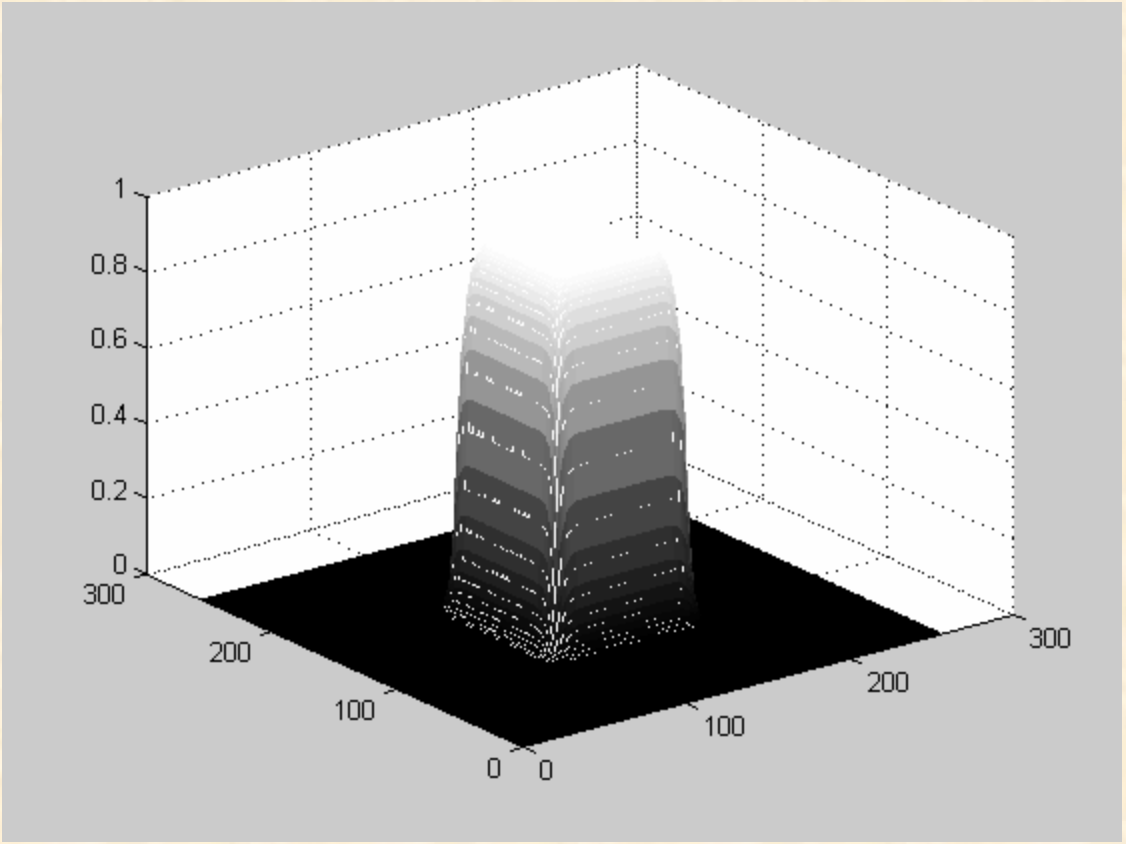


Low-pass filtering



Butterworth filter

n = 1



P. Strumiłło, M. Strzelecki

Low-pass Butterworth filter - examples

n = 1

D_0=10

D_0=70

no ringing

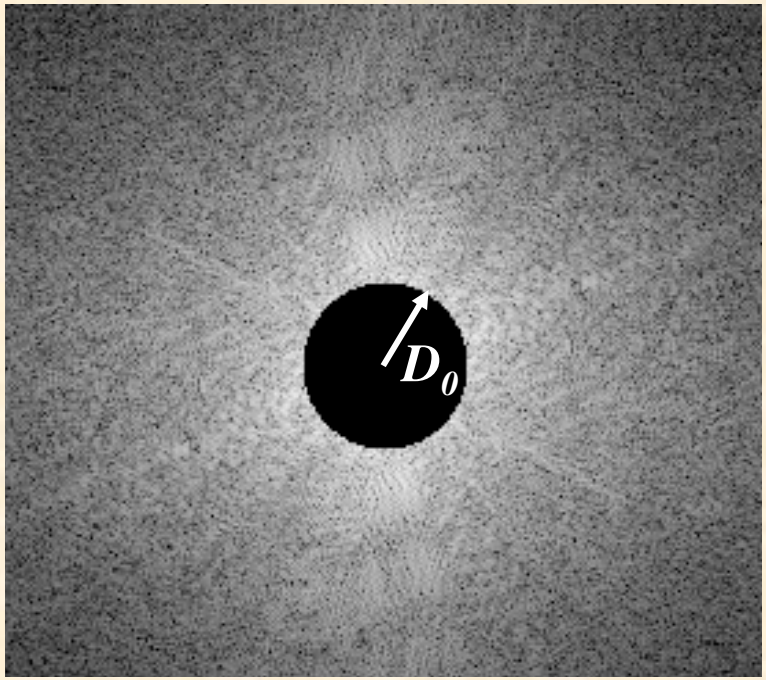
D_0=30

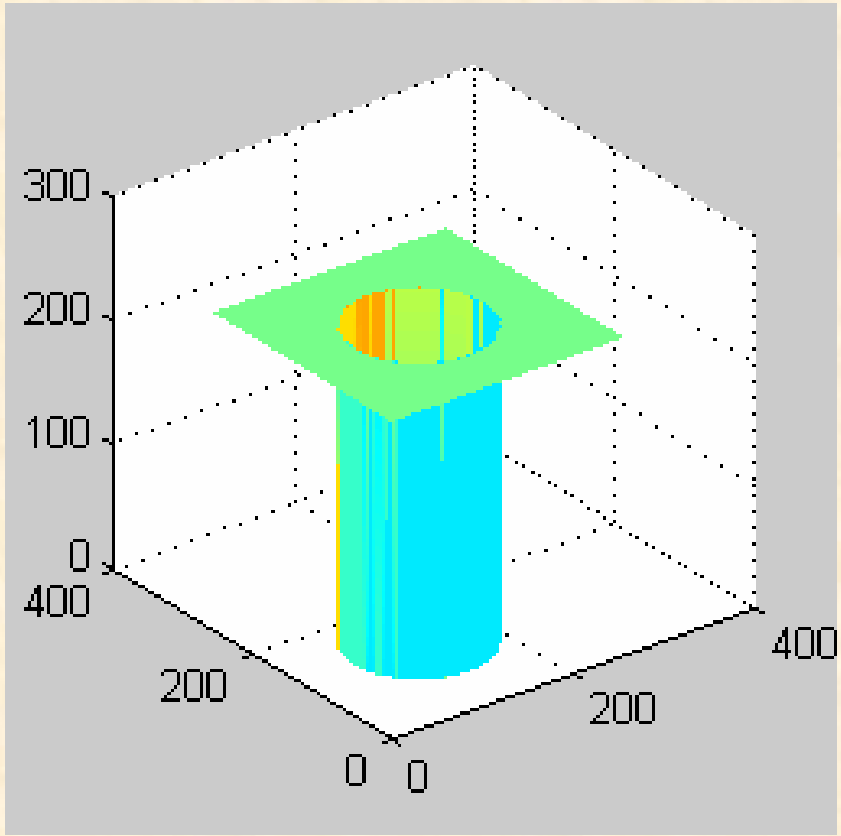
P. Strumiłło, M. Strzelecki

Ideal high-pass filter

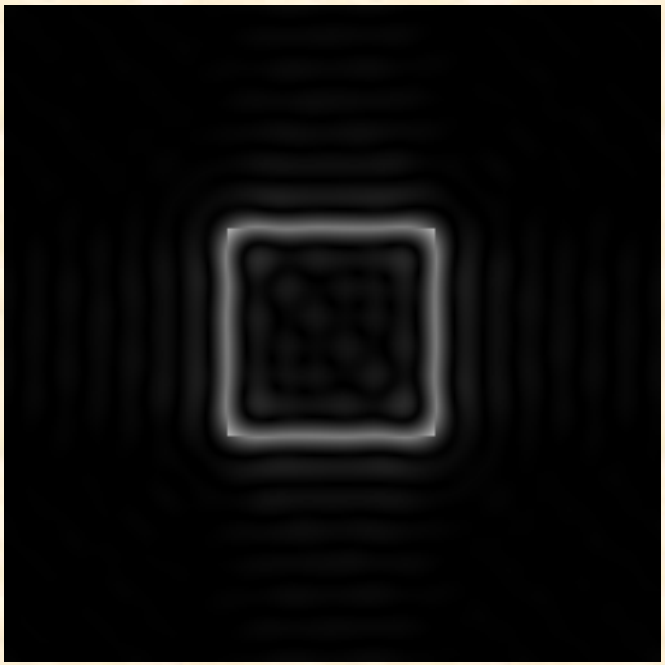
$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \sqrt{u^2 + v^2}$$

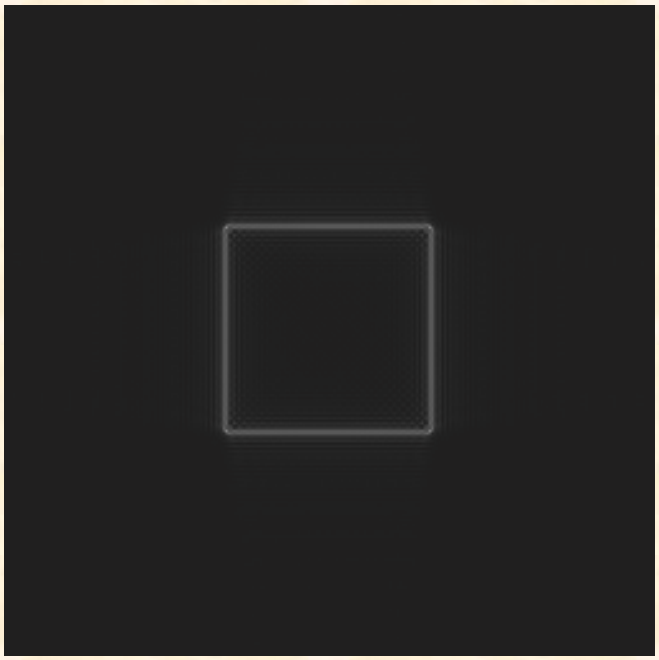




Ideal high-pass filter - example

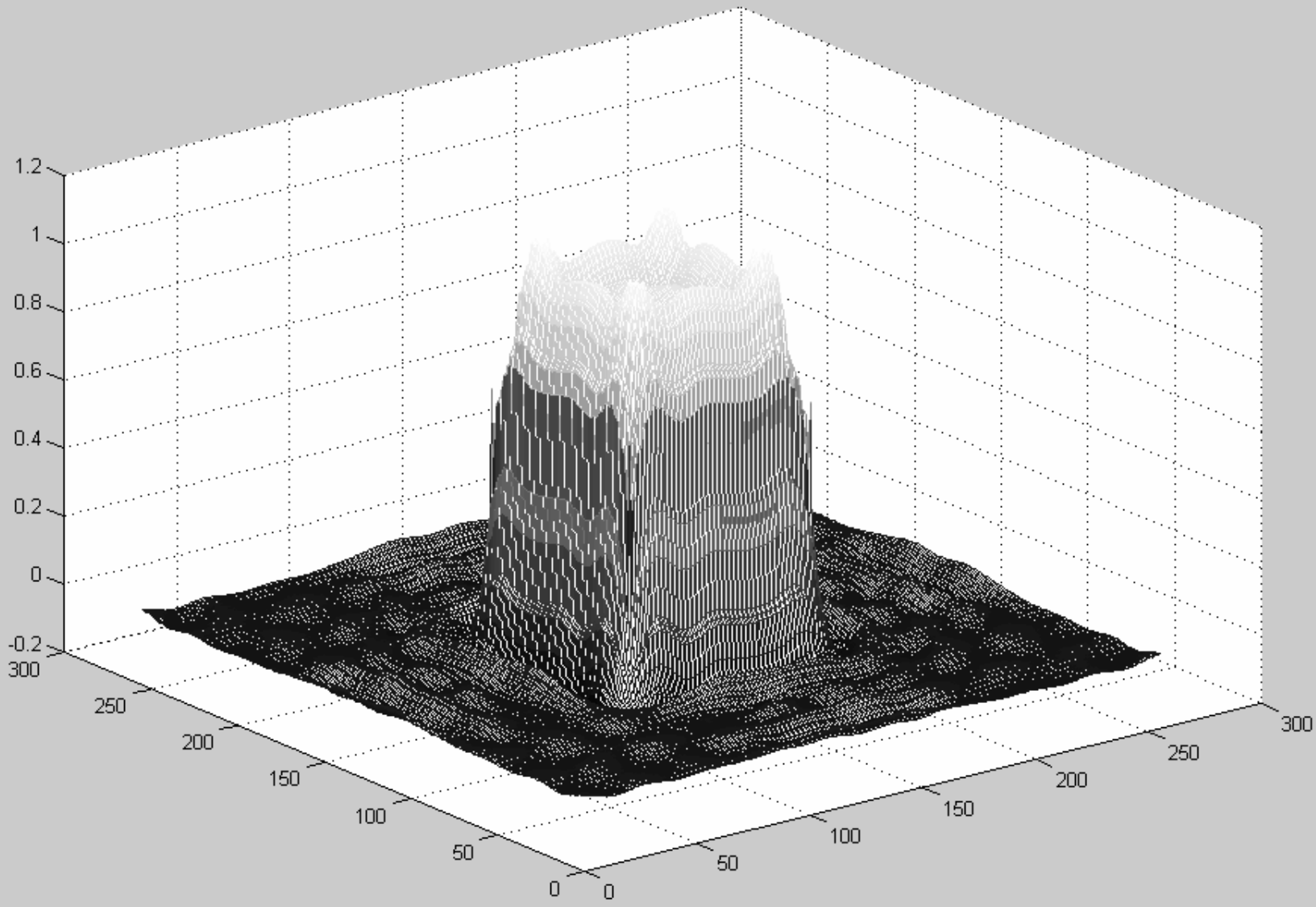


$D_0=10$



$D_0=70$

Image after ideal high-pass filtering, D_0=10



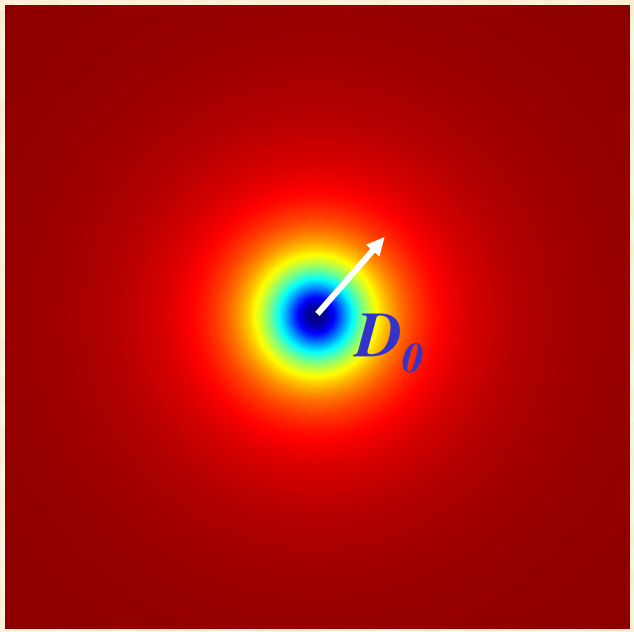
High-pass Butterworth filter 2

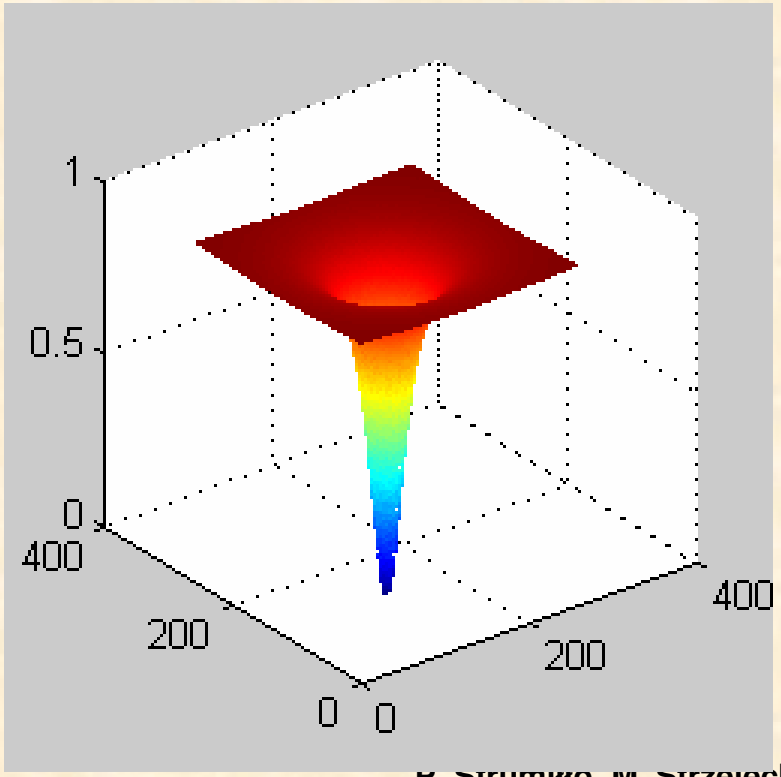
$$H(u,v) = \frac{1}{1 + (\sqrt{2} - 1)[D_0 / D(u,v)]^{2n}}$$

$$D(u,v) = \sqrt{u^2 + v^2}$$

$$n = 1,2,\ldots$$

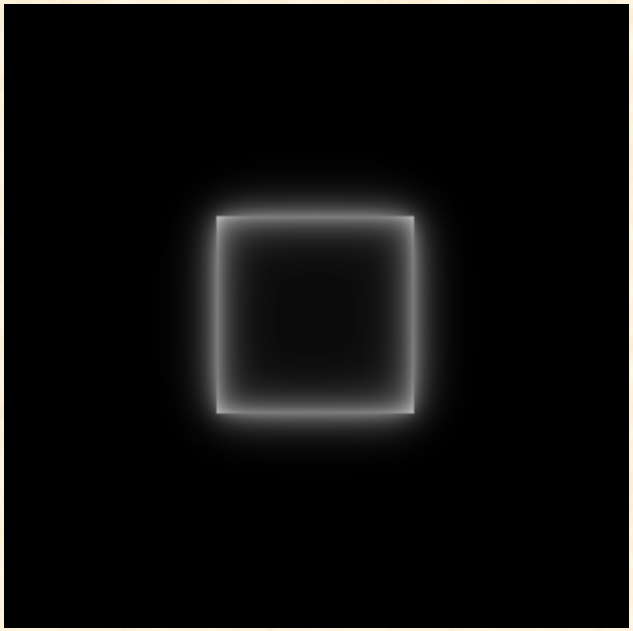
n - filter order



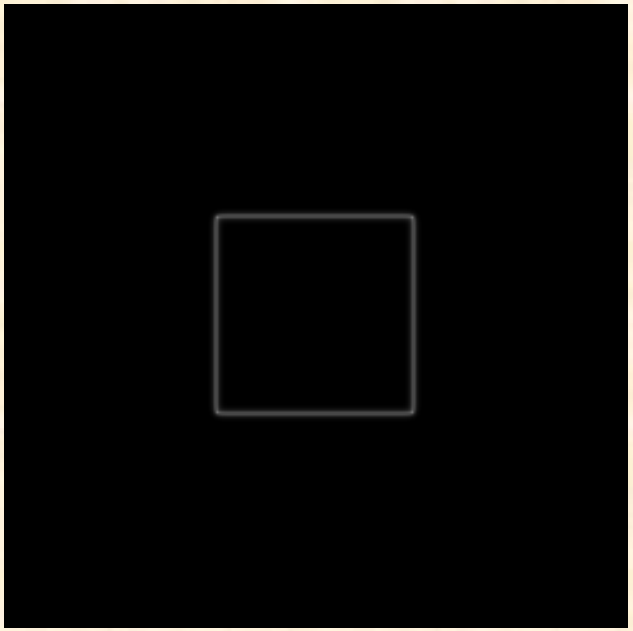


High-pass Butterworth filter - examples

n = 1



D_0=10



D_0=70

High-pass Butterworth filter - examples

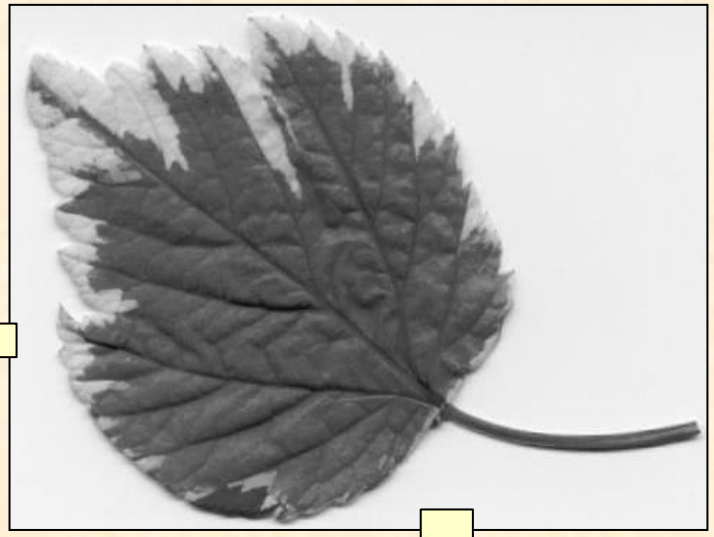
n = 1

D_0=10

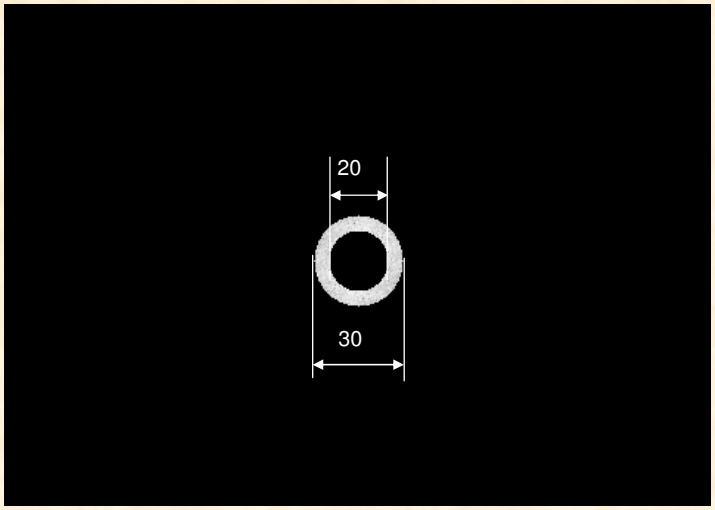
D_0=70

Band-pass filter

The filter selects narrow band of image frequencies



FFT



IFFT

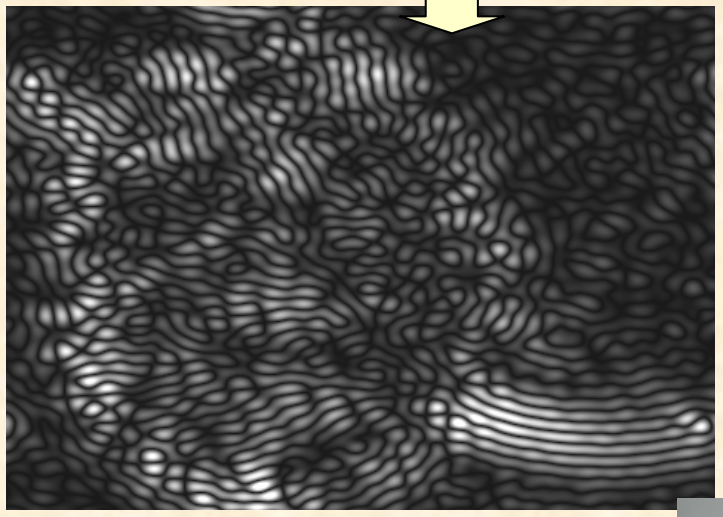
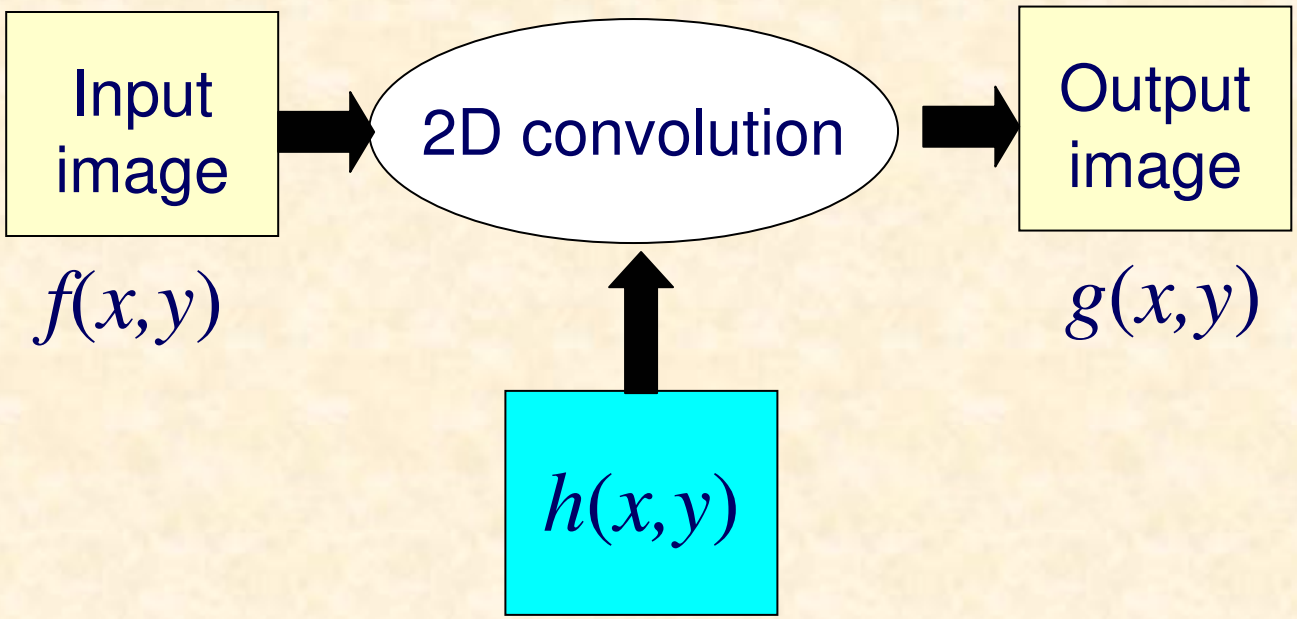
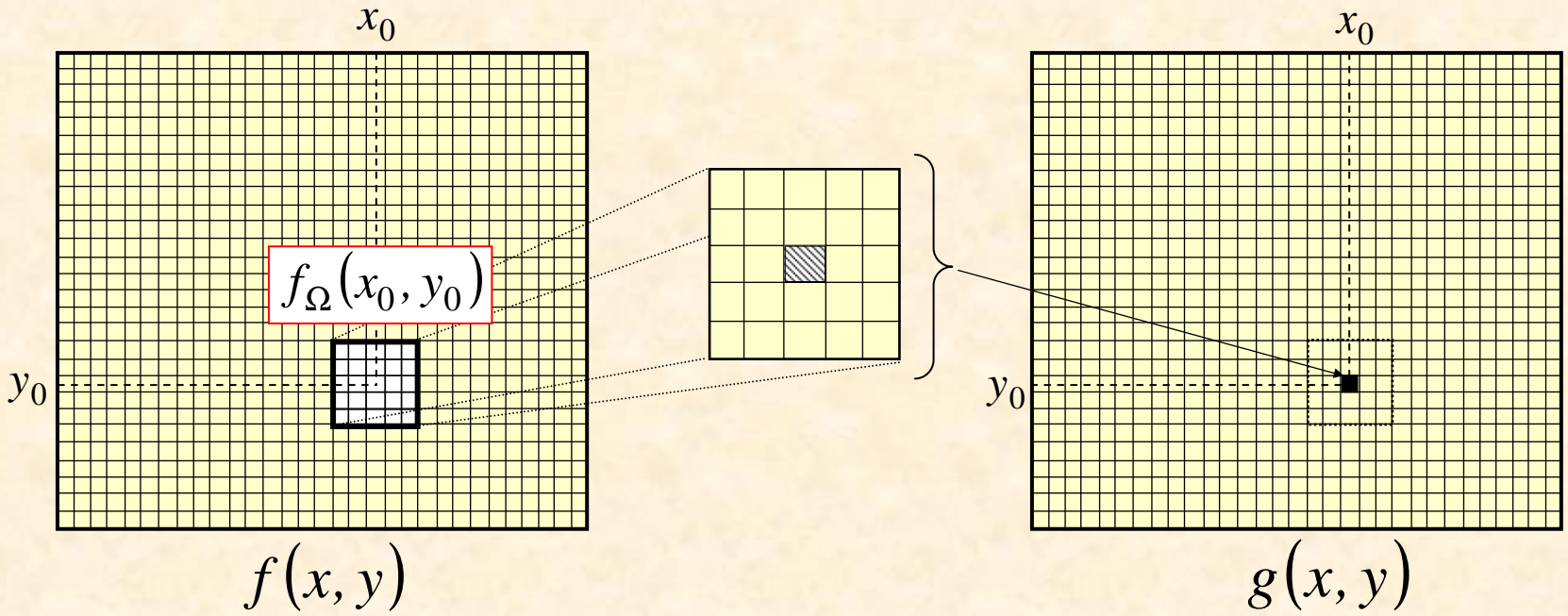


Image filtering in spatial domain



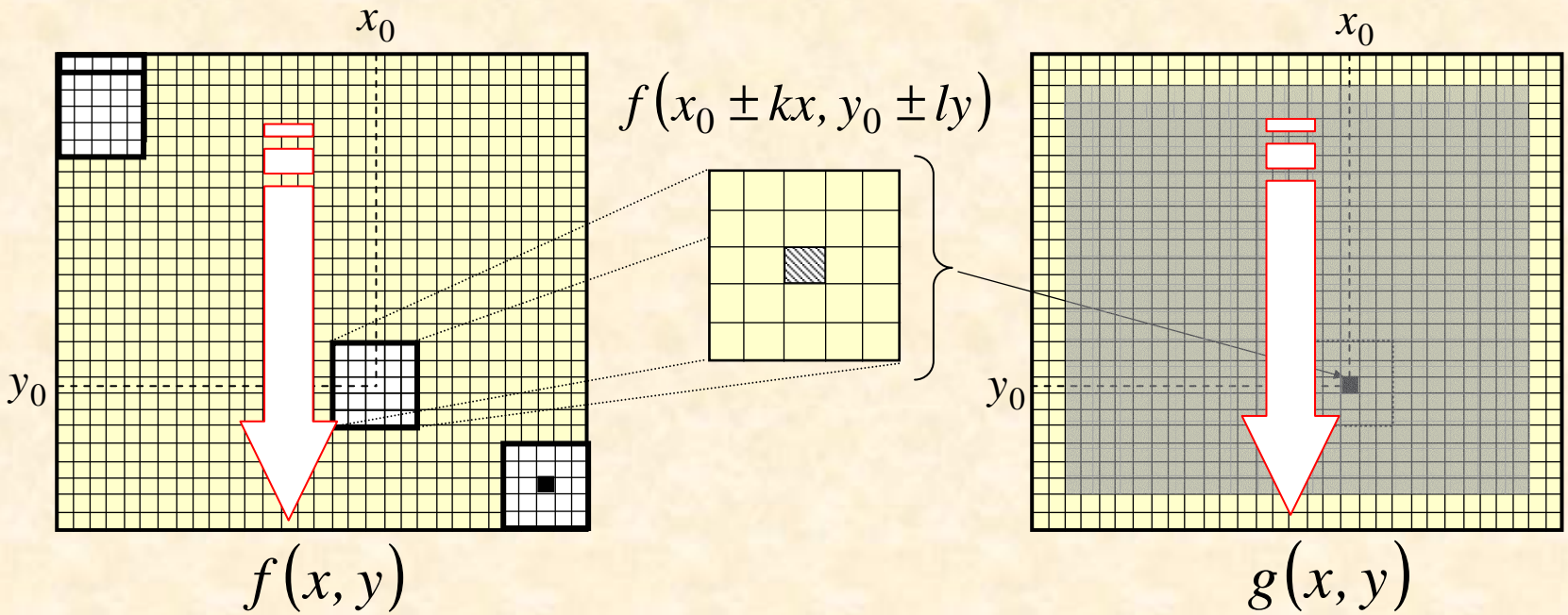
$$g(x,y) = h(x,y) ** f(x,y) =$$
$$= IF\{ H(u,v)\, F(u,v)\}$$

Image filtering in spatial domain

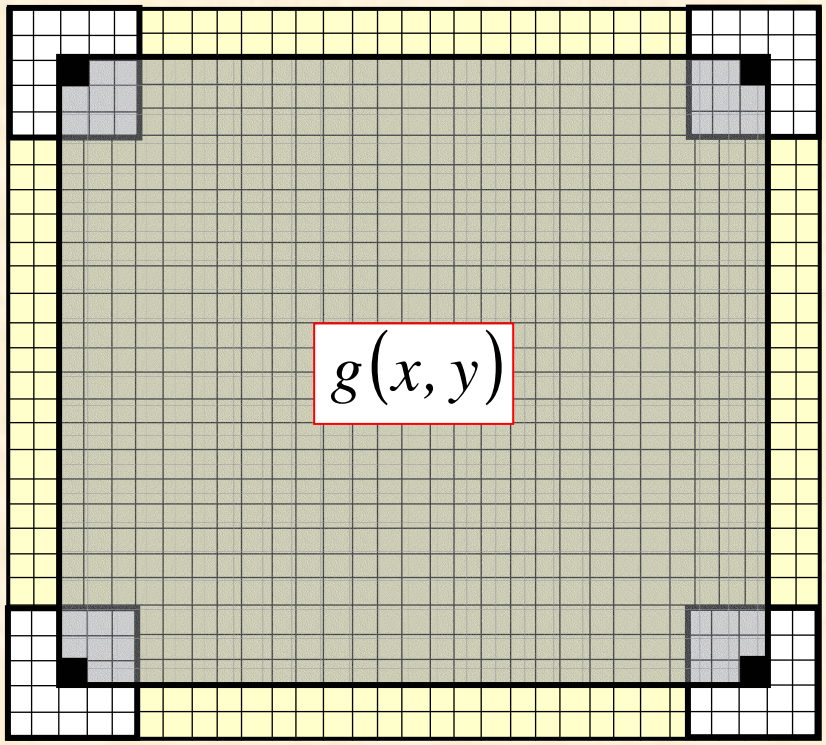


The filtering result depends on the processed pixel neighbourhood. This neighbourhood is call a **filtering window** (or a filtering mask).

Image filtering in spatial domain



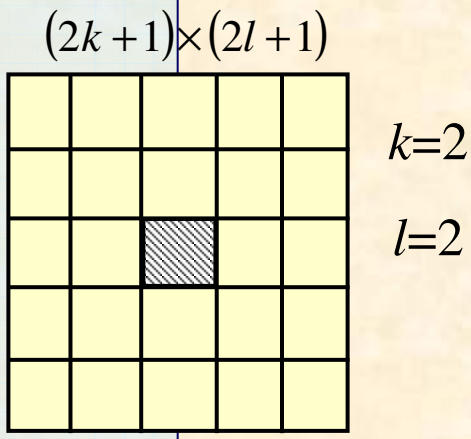
Boundary elements



The result of filtering an image of size M x N for the filtering window size $(2k+1)$x$(2l+1)$ is an image of size $(M-2k)$x$(N-2l)$

Filtering algorithm

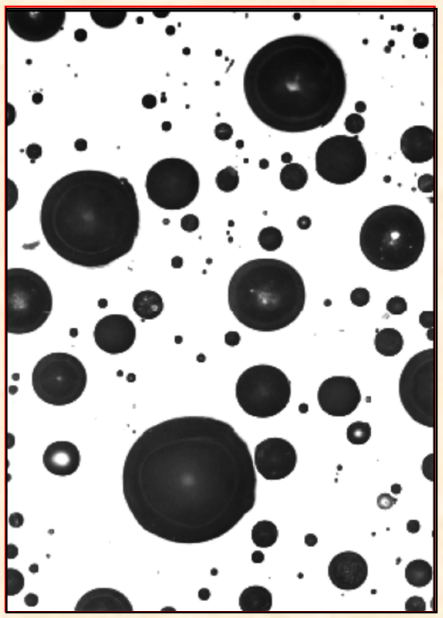
```
f: array[0..M-1,0..N-1] of byte; //source image
g: array[0..M-1,0..N-1] of real; //filtered image
...
for x:=k to M-k-1 do
  for y:=l to N-l-1 do
            g(x,y):=P(f(x,y)); //perform local computations
  end;
end;
```



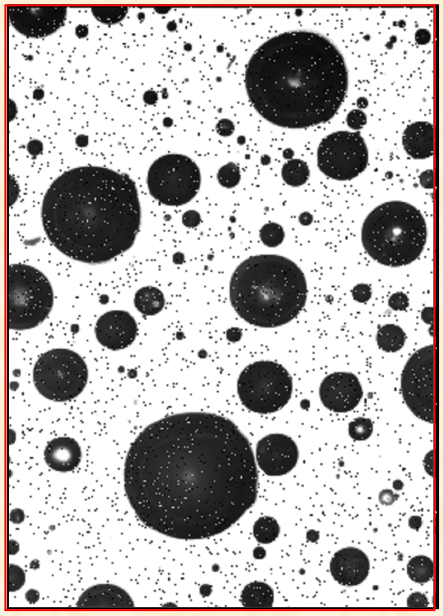
Operator *P* [.] can be linear or nonlinear.

Correspondingly we define **linear** or **nonlinear** image filtering.

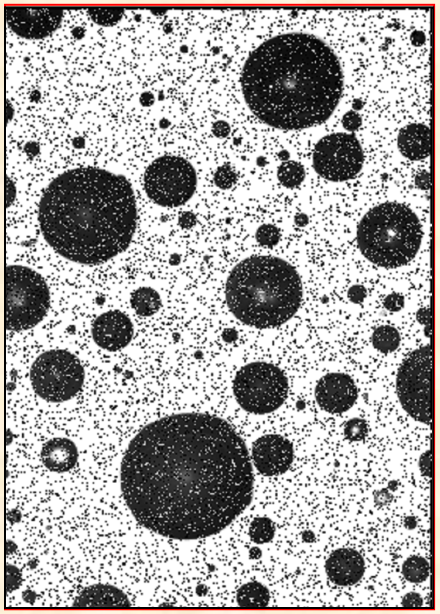
Salt and pepper noise



Source



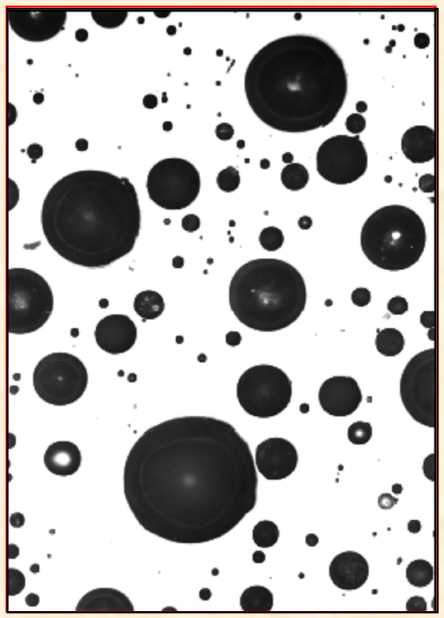
0.05, i.e. 5% of distorted pixels



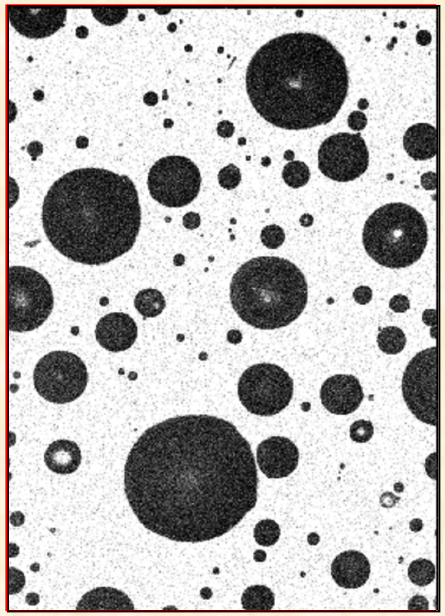
0.5, i.e. 20% of distorted pixels

Randomly selected pixels are assigned minimum or maximum value randomly.

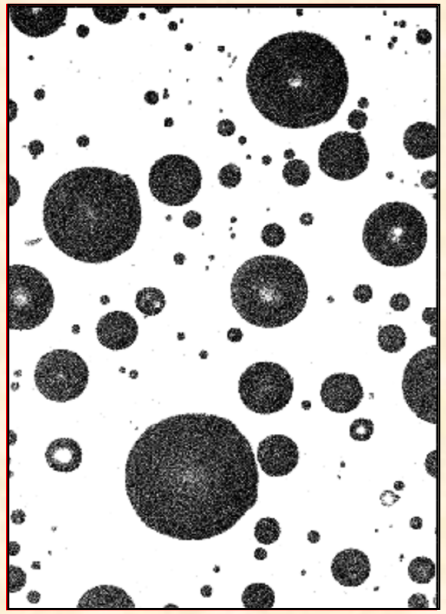
Gaussian noise



Source



σ^2= 0.05



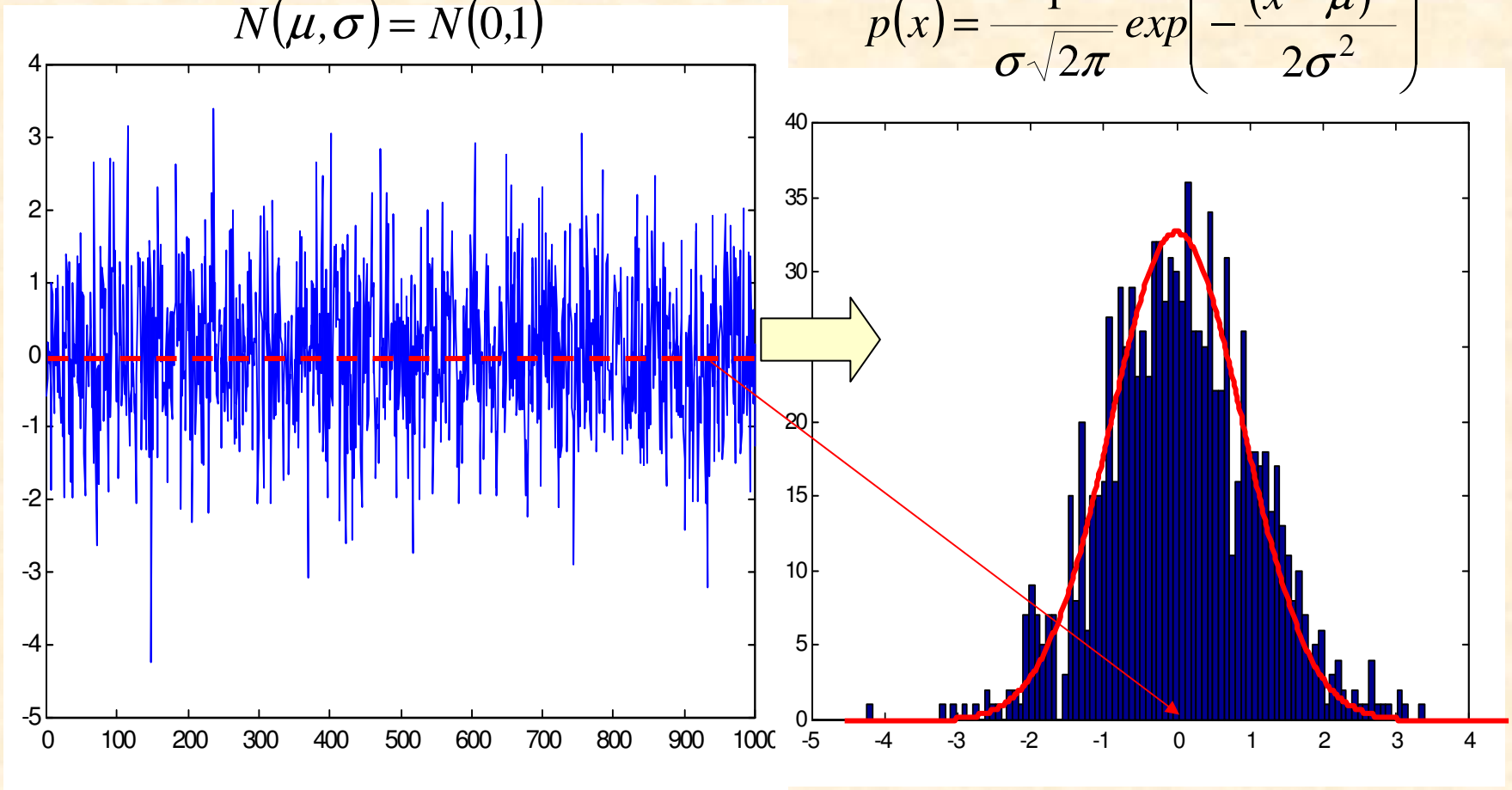
σ^2=0.2

For each image pixel: $g(x,y) = f(x,y) + n(x,y)$

The Gauss distribution

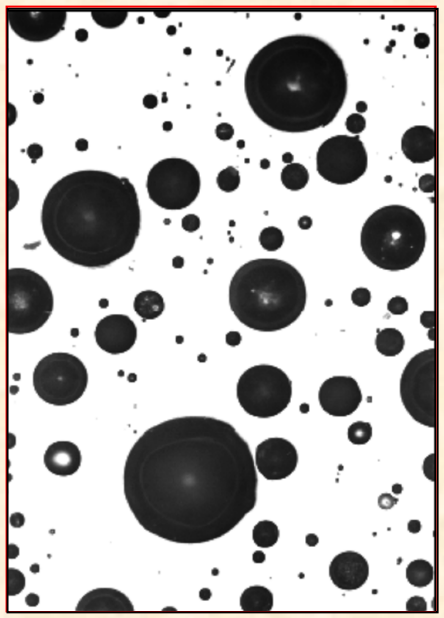
$$N(\mu,\sigma) = N(0,1)$$

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

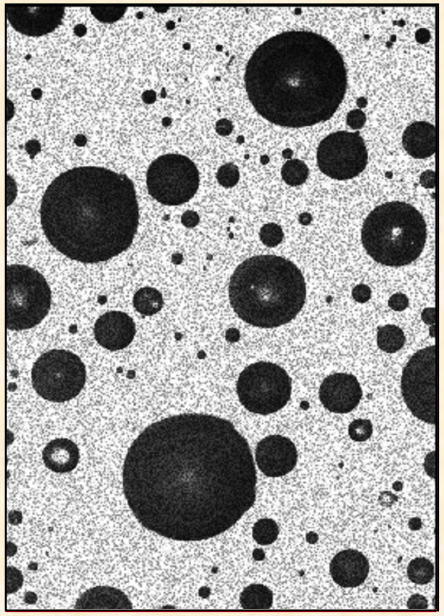


P. Strumiłło, M. Strzelecki

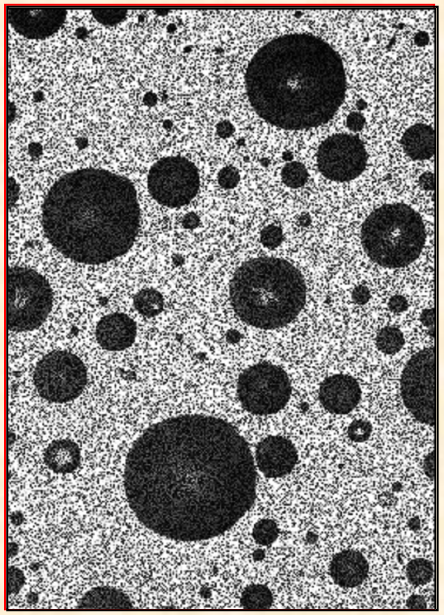
Multiplicative noise



Source



$\sigma^2 = 0.05$



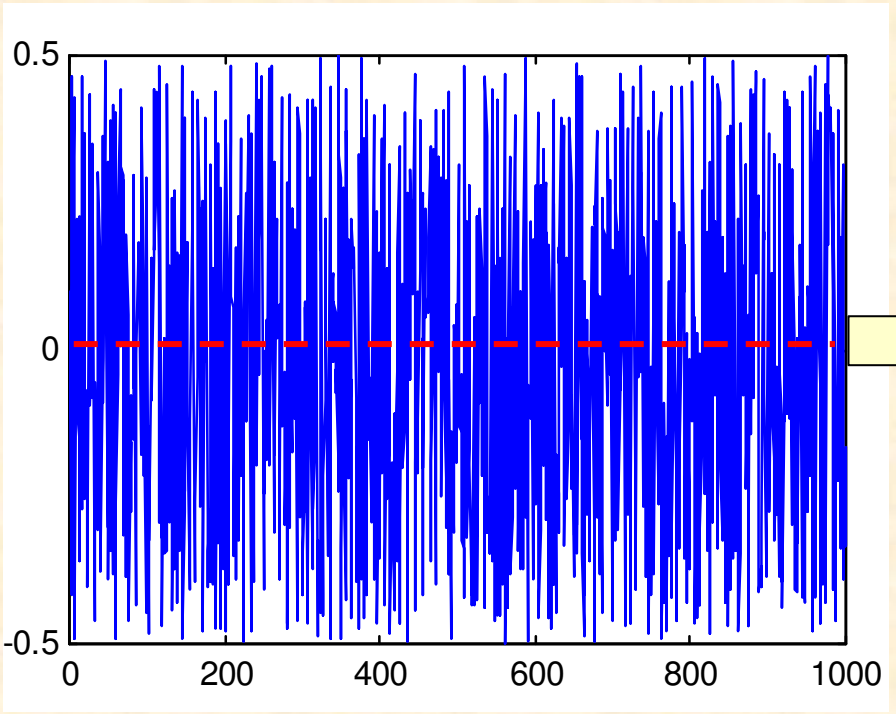
$\sigma^2 = 0.2$

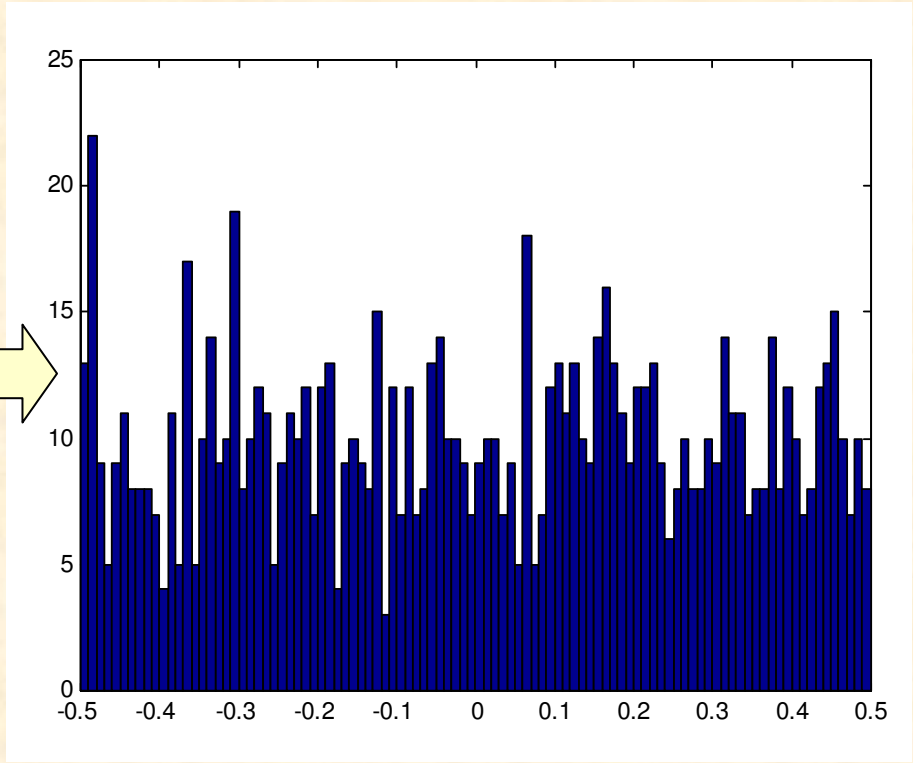
For each image pixel: $g(x,y) = f(x,y) + n(x,y) \cdot f(x,y)$

Uniform noise

P. Strumiłło, M. Strzelecki

Uniform noise





Linear filtering

Operator *P* [.] is a linear function of neighbour pixels.

Example – an averaging filter

$$g(x,y) = P[f_\Omega(x,y)] = \frac{1}{(2k+1)(2l+1)} \sum_{x,y \in f_\Omega} f(x,y) \quad = \quad h(k,l) = \frac{1}{25} \times$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Filtering mask

Why $\sum$ coefficients = 1?

Image filtering – calculations

Multiplication of filter coefficient by the correcsponding image pixels

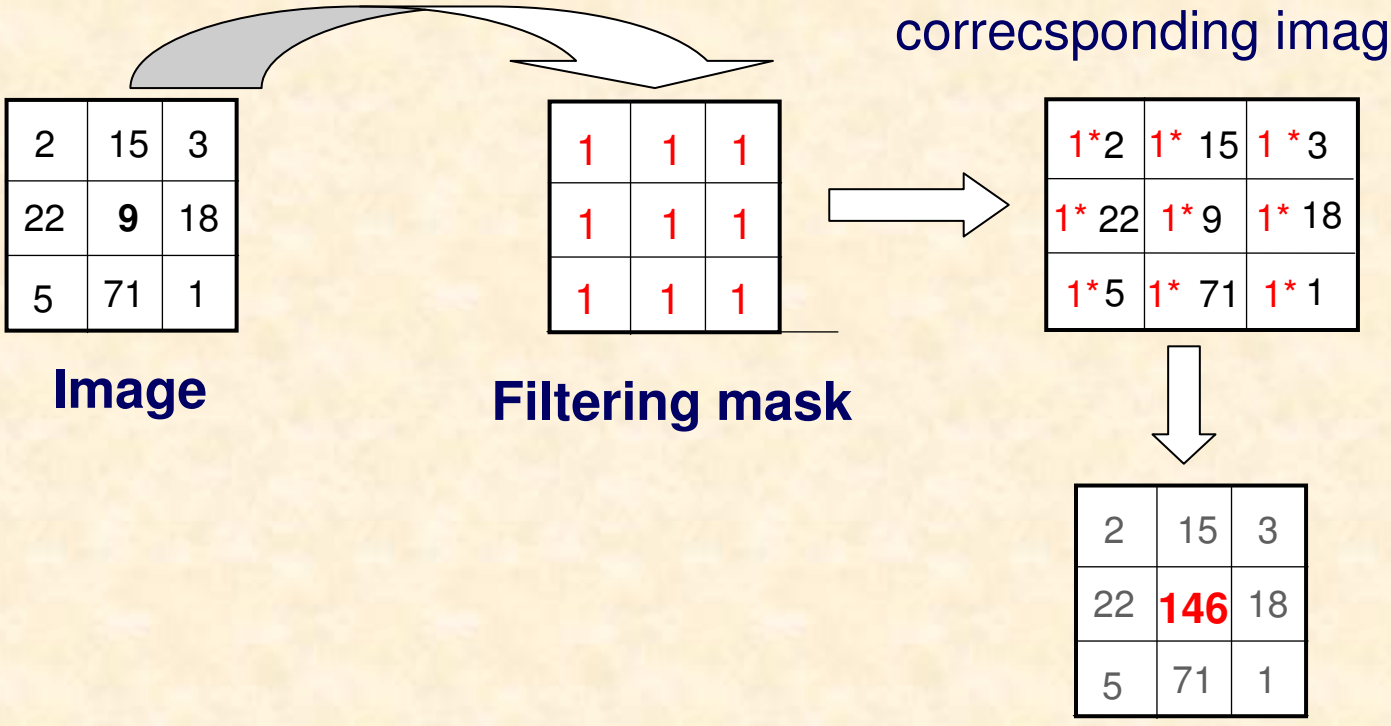


Image filtering – the algorithm

```
f, g : array[0..N-1, 0..N-1] of byte;
{ size2 – half size of the mask}
h : array[-size2..size2,-size2..size2] of integer;
...
for i:=1 to N-2 do  for j:=1 to N-2 do
        begin
           g[i,j]:=0;
           for k:=-size2 to size2 do  for l:=-size2 to size2 do
                 g[i,j]:=g[i,j] + f[i+k,j+l] * h[i+k,j+l];
        end;
...
```

Range check g[i,j] !!!

Convolution in 2D

Linear filtering of an image is given by performing 2D convolution of an image with the filtering mask

$$g(x,y) = h(x,y) * f(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k,l) f(x-k, x-l)$$

Filtering mask

Image

Convolution in 2D

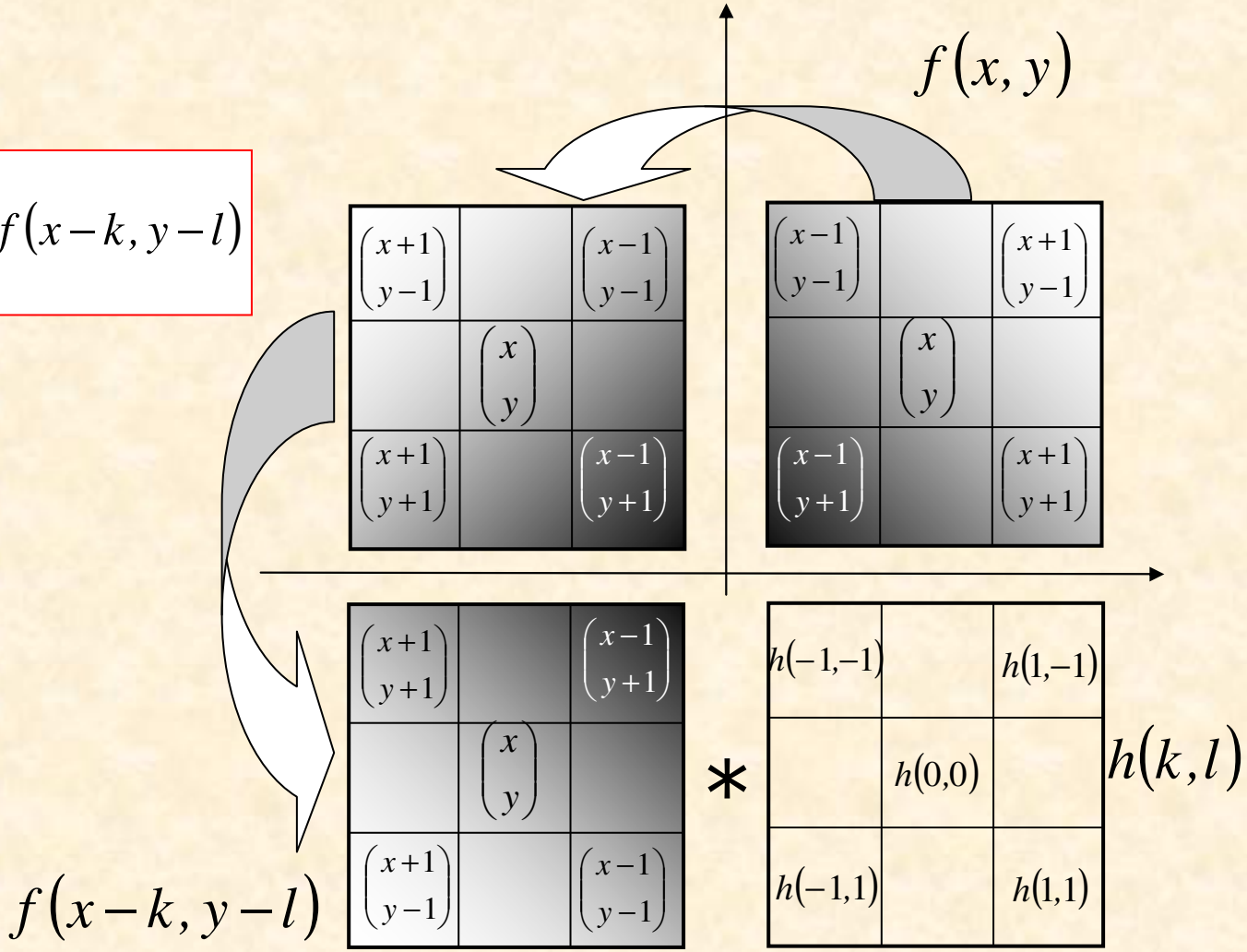
For a 3x3 filtering mask at each (*x,y*) image pixel coordinate the following calculations take place:

$$g(x,y)=\sum_{k=-1}^{1}\sum_{l=-1}^{1}h(k,l)f(x-k,y-l)=$$
$$=h(-1,-1)f(x+1,\ y+1)+h(0,\ -1)f(x,y+1)+h(1,-1)f(x-1,y+1)+$$
$$+h(-1,\ 0)f(x+1,\quad y)+h(0,\quad 0)f(x,\quad y)+h(1,\ 0)f(x-1,\quad y)+$$
$$+h(-1,\quad 1)f(x+1,y-1)+h(0,\quad 1)f(x,\ y-1)+h(1,\ 1)f(x-1,y-1)$$

eg. for a 512x512 image 510*510*9 = 2 340 000 multiplications and additions need to be calculated

Convolution in 2D

$$g(x,y)=\sum_{k=-1}^{1}\sum_{l=-1}^{1}h(k,l)f(x-k,y-l)$$

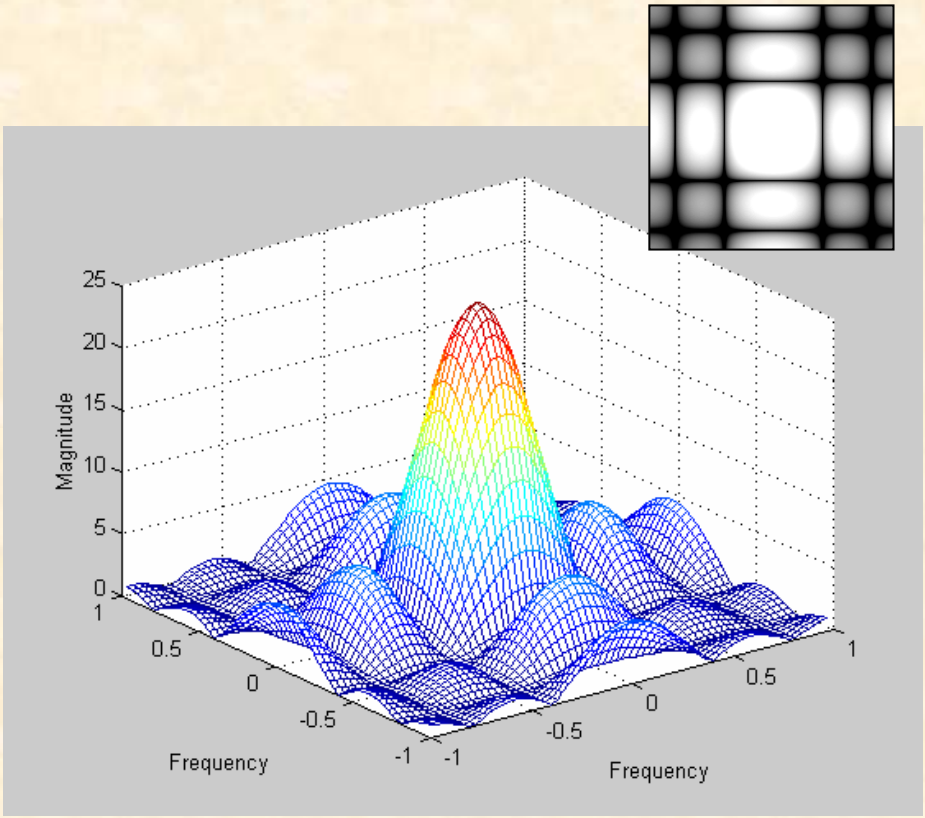


Averaging = low pass filter

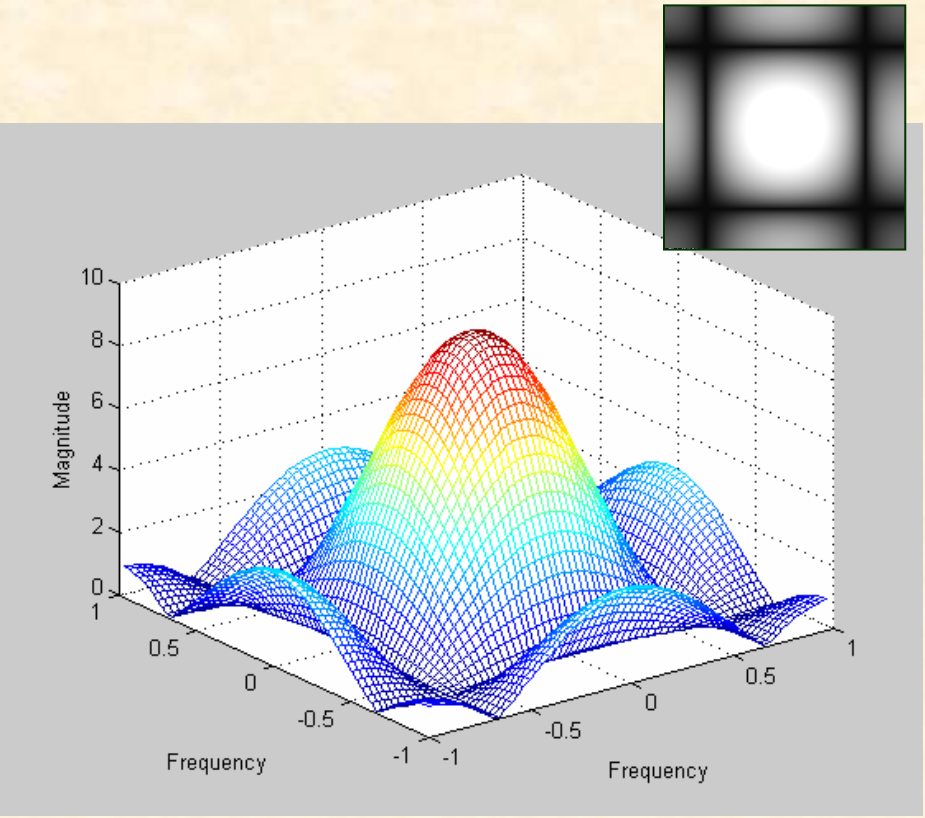
$$h_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h_2 = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Can one use mask of even size ?

Frequency characteristics of low pass filters



for 5x5 mask



for 3x3 mask

P. Strumiłło, M. Strzelecki

Low-pass filtering the image

Source image

3x3 mask

5x5 mask

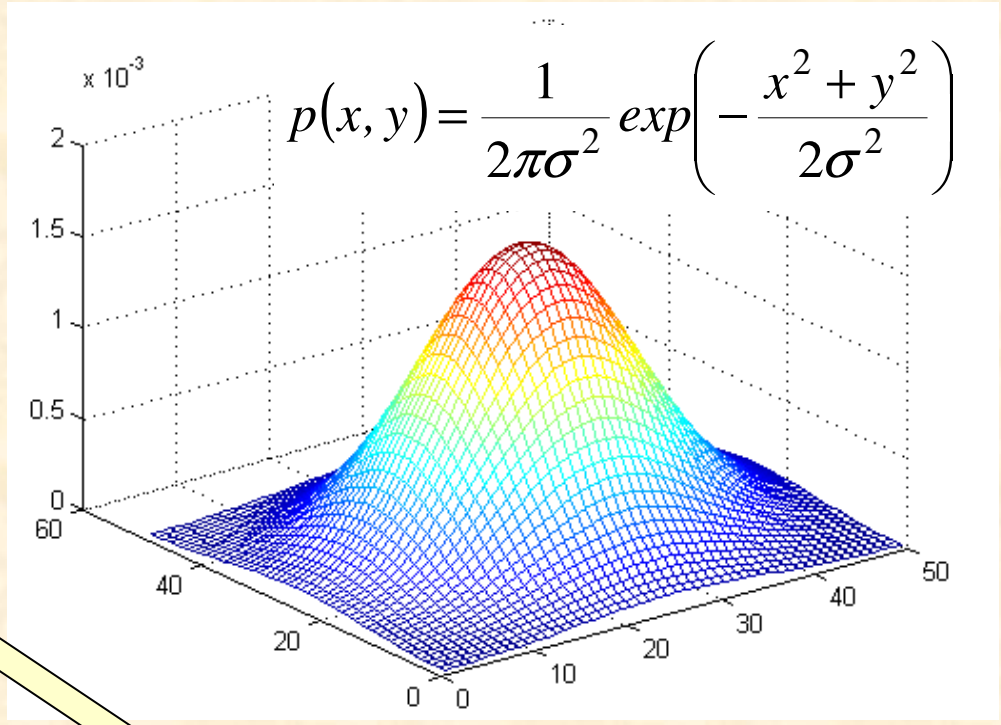
The Gaussian filter

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

1	2	1
2	4	2
1	2	1

$$\frac{1}{96}\times$$

0	1	2	1	0
1	6	10	6	1
2	10	16	10	2
1	6	10	6	1
0	1	2	1	0



Gaussian filters on a discrete grid

Integer coefficients require less computations

Image filtering using the Gaussian filter

source image

filtered image

Smoothing filters - comparison

Source

Gaussian 5x5, σ=1

Averaging 5x5

Image low-pass filters - examples

Image distorted by the Gaussin noise N(0, 0.01)

for grayscale <0,1>

Low pass filter 3x3

Gaussian filter 3x3

Butterworth filter D_0=50

Image low-pass filters - examples

Image distorted by the Gaussian noise N(0, 0.01)

low-pass filter 5x5

Gaussian filter 5x5

Butterworth filter D_0=30

Image low-pass filters - examples

Image distorted by the Gaussian noise N(0, 0.002)

Low pass filter 3x3

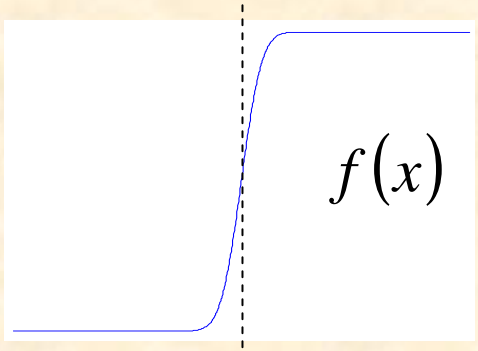
Gaussian filter 3x3

Butterworth filter D_0=50

Image sharpening filters

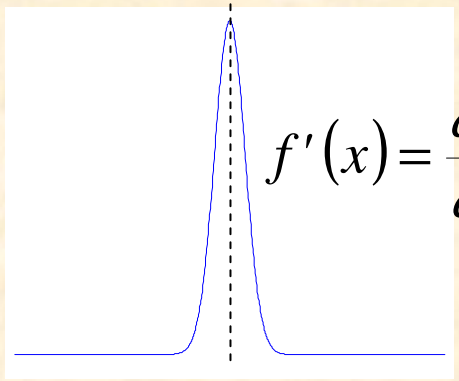
For $\Delta x=1$:

$$f'(x) \cong \frac{\Delta f}{\Delta x} = f(x,y) - f(x+1,y)$$



Corresponds to a filter mask:

1	-1

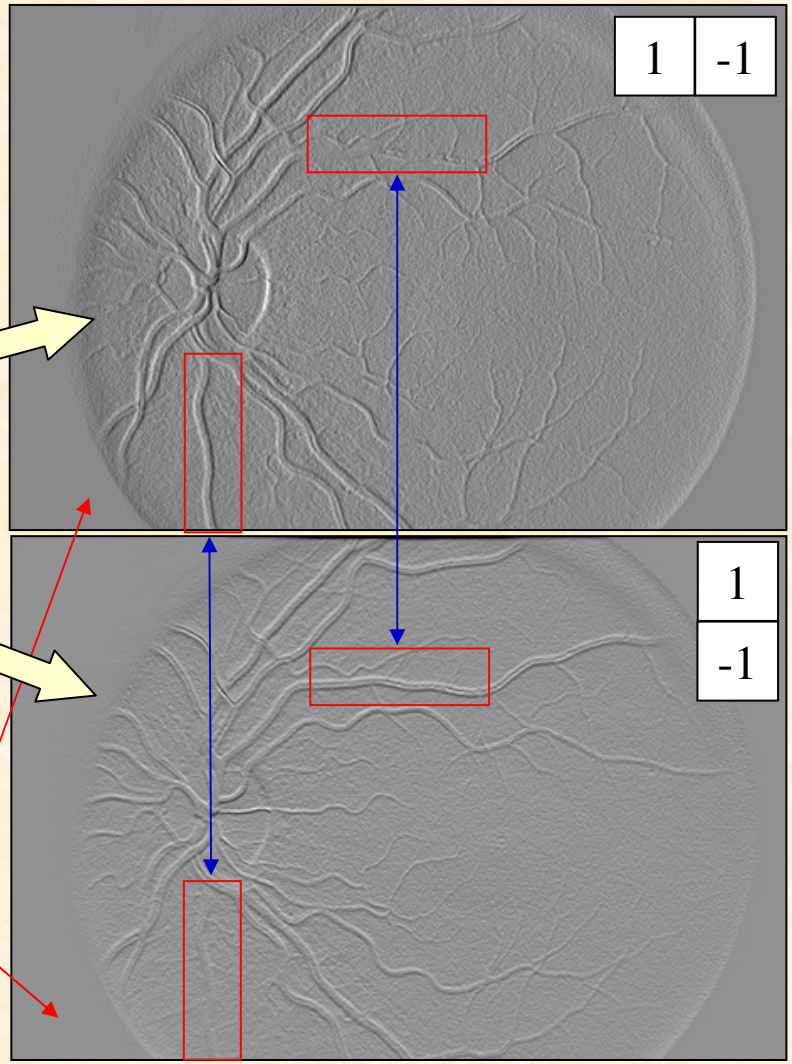


Similarily for y ($\Delta y=1$) :

1
-1

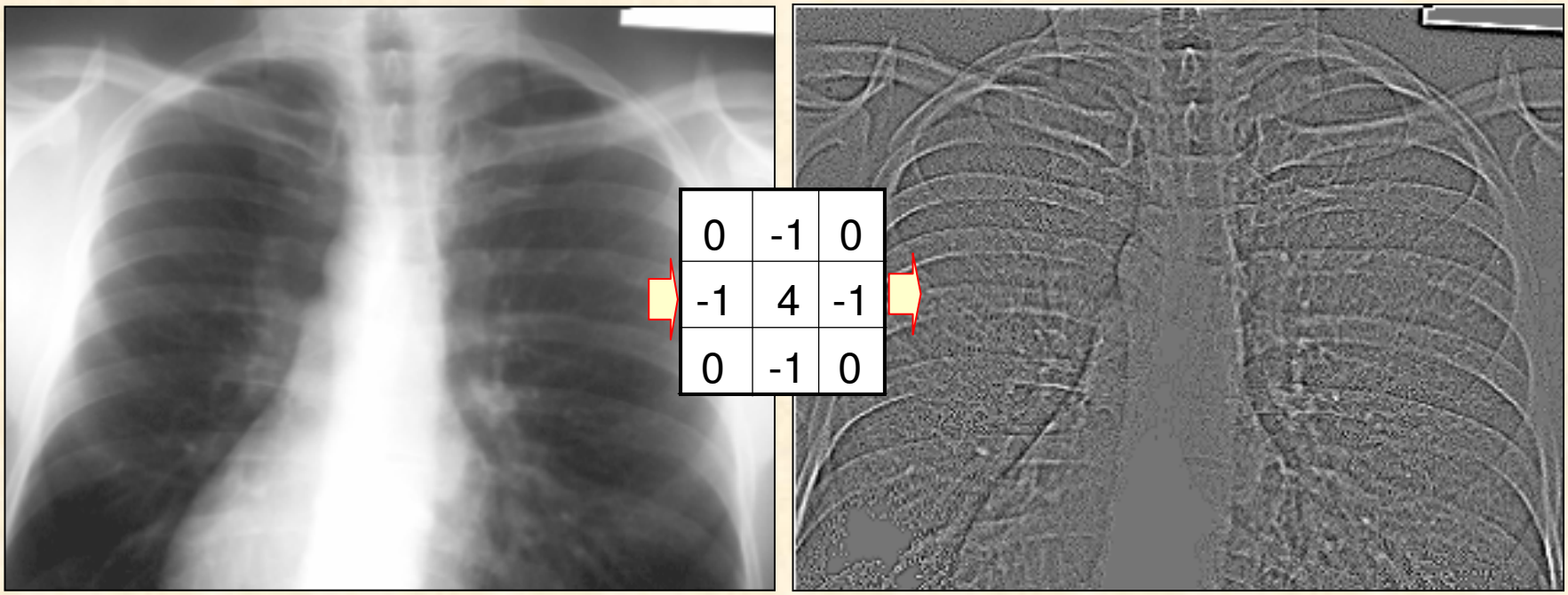
First order gradient filters

An image of retinal arteries



Gradient images

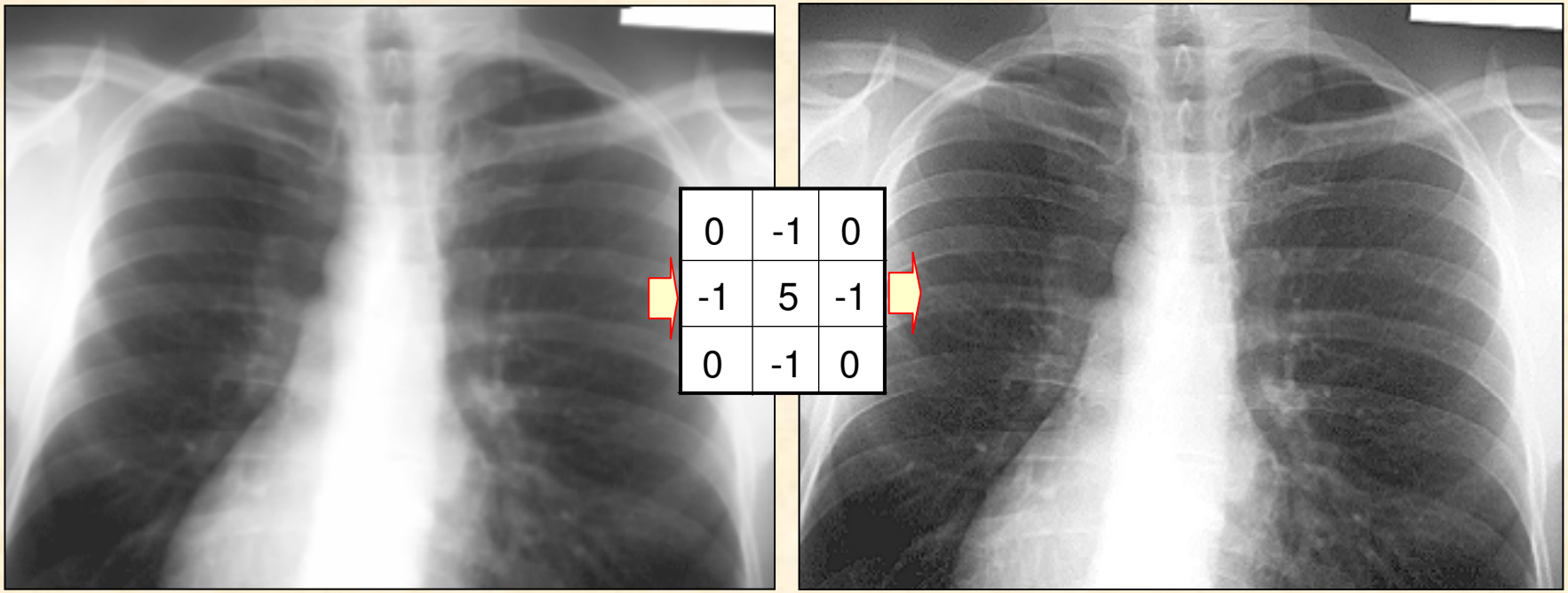
Second order gradient filters



How to work out this filter mask?

Hint: compute second order image differences along columns and rows and add the results

Second order image sharpening filter



An original image was added to the second order gradient mask:

0	-1	0
-1	4	-1
0	-1	0

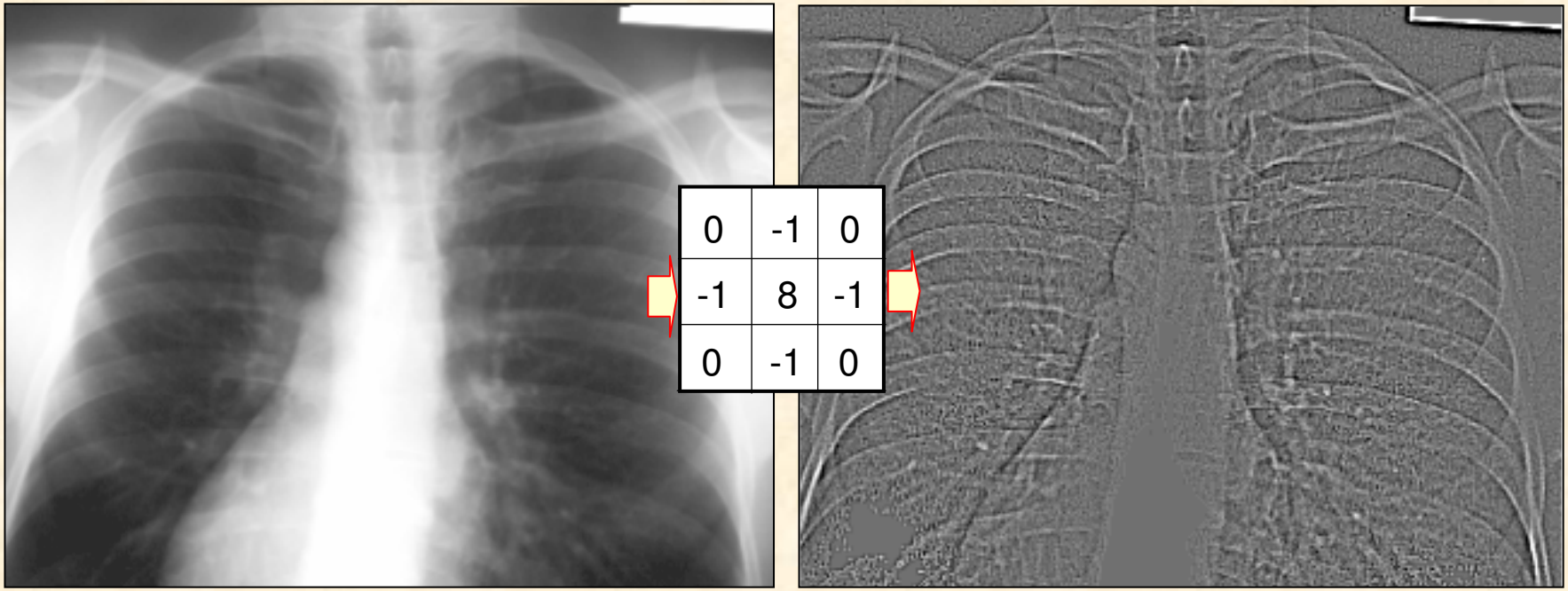
+

0	0	0
0	1	0
0	0	0

=

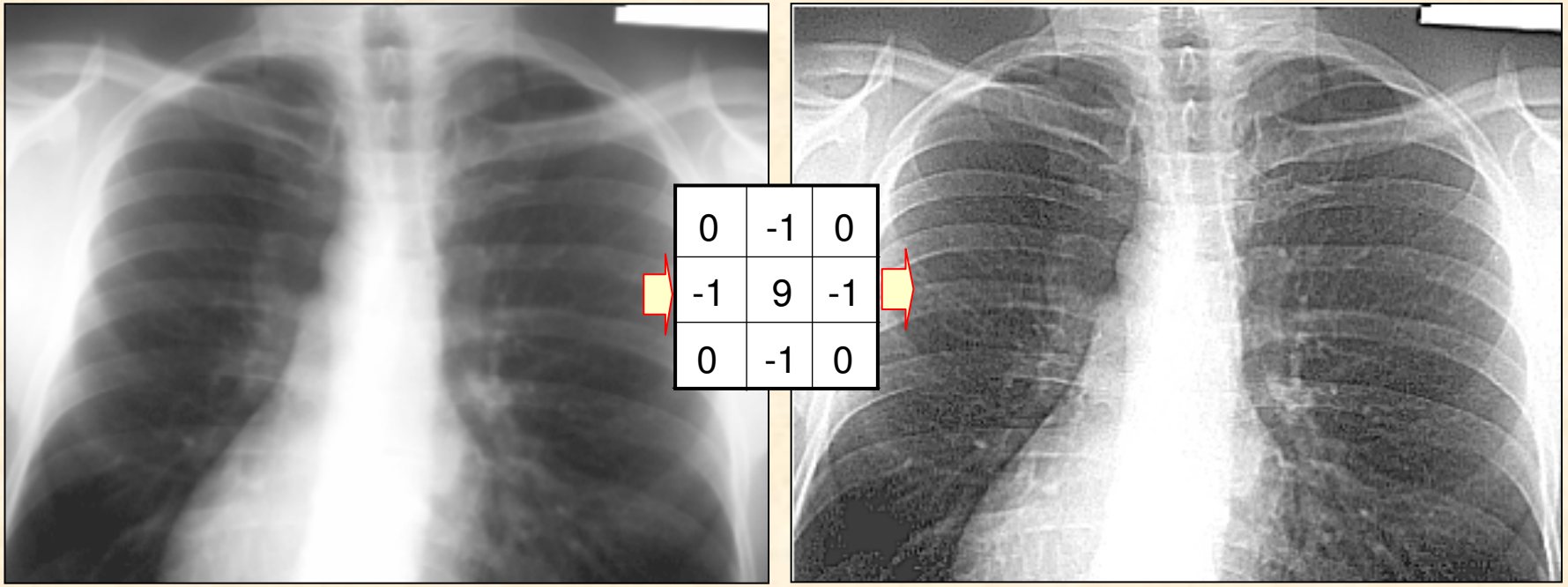
0	-1	0
-1	5	-1
0	-1	0

The Laplace filter



Please compute coefficients of this filter

The Laplace filter



An original image was added to the second order gradient mask:

-1	-1	-1
-1	8	-1
-1	-1	-1

+

0	0	0
0	1	0
0	0	0

=

-1	-1	-1
-1	9	-1
-1	-1	-1

Comparison of sharpening filters

$$h'_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$h'_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

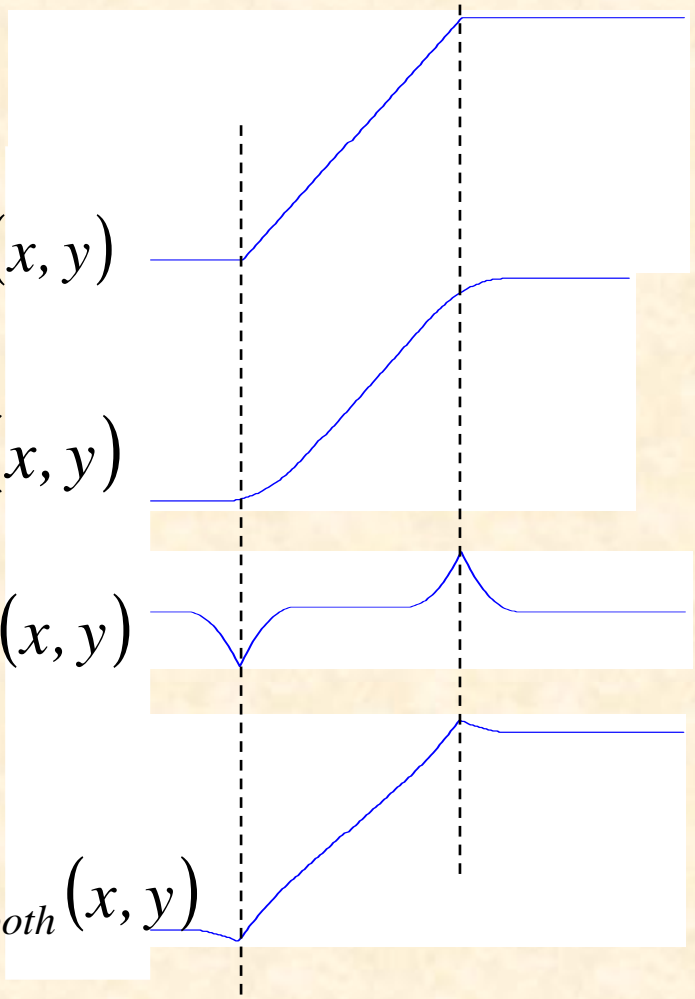
Unsharp masking filter

$$f(x,y)$$

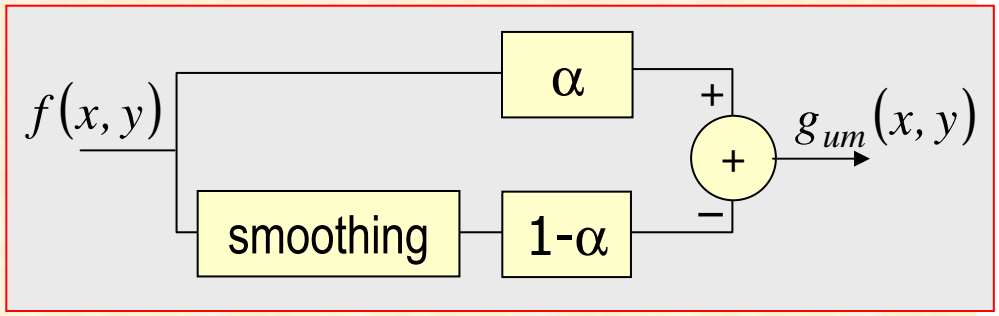
$$f_{smooth}(x,y)$$

$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

$$g_{um}(x,y) = g(x,y) + f(x,y) = 2f(x,y) - f_{smooth}(x,y)$$

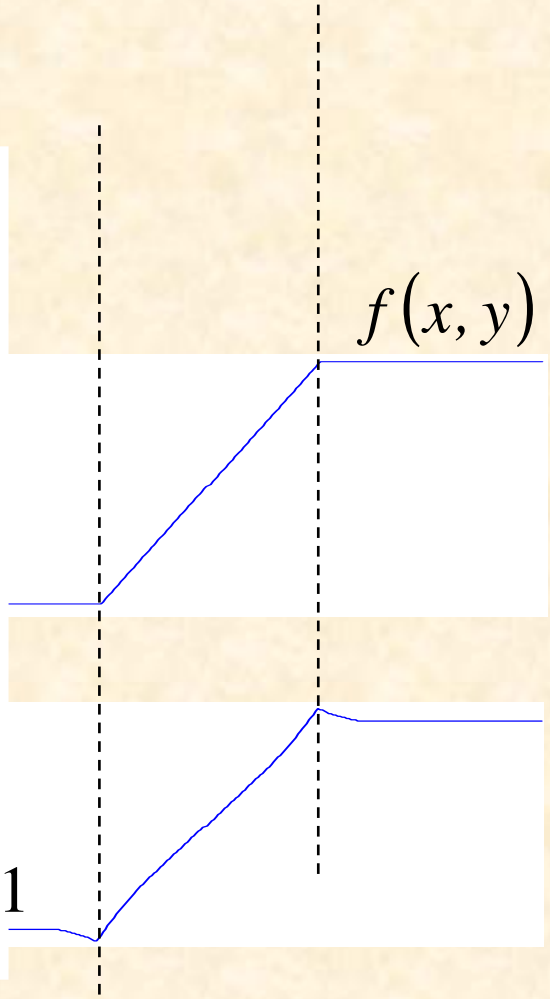


Unsharp masking filter



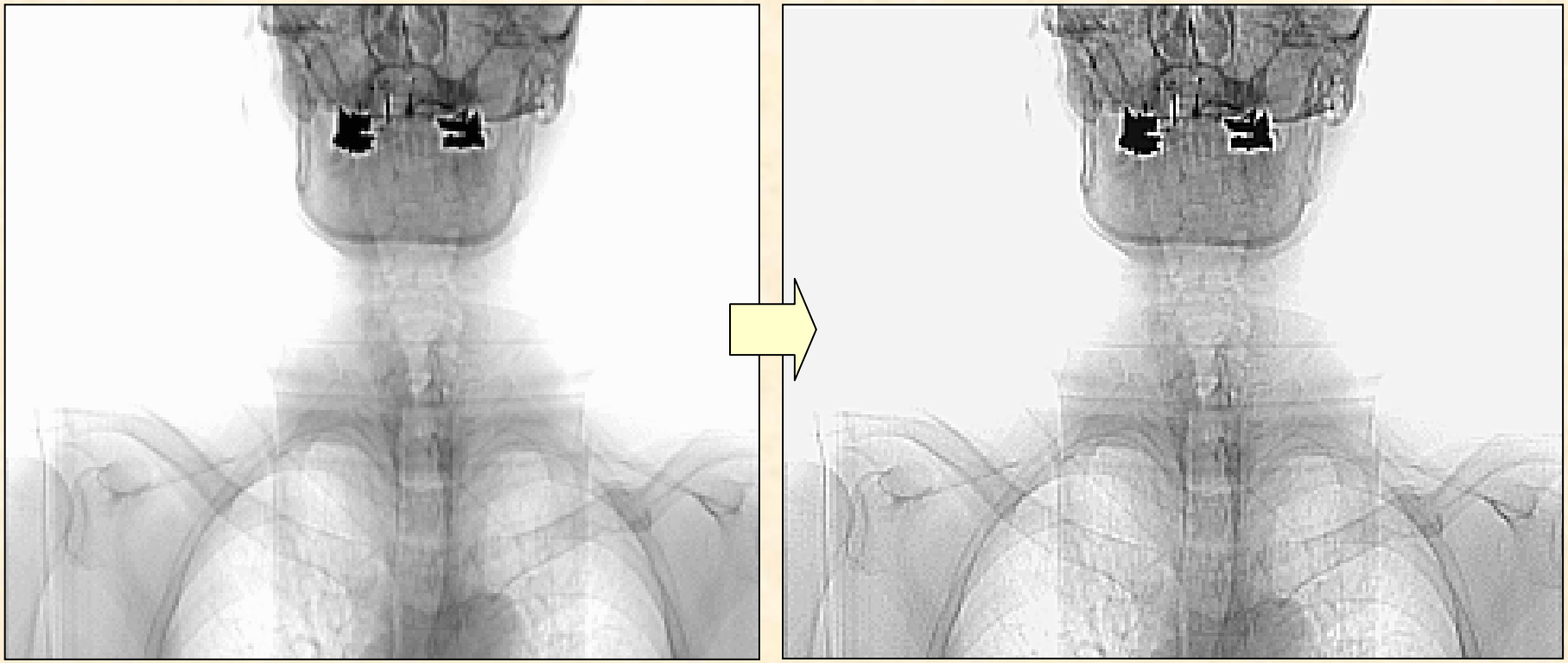
Parametr α decides about the sharpening strength.

$$g_{um}(x,y) = \alpha f(x,y) - (1-\alpha) f_{smooth}(x,y),\ \alpha < 1$$



P. Strumiłło, M. Strzelecki

Unsharp masking filter



α=2/3

P. Strumiłło, M. Strzelecki

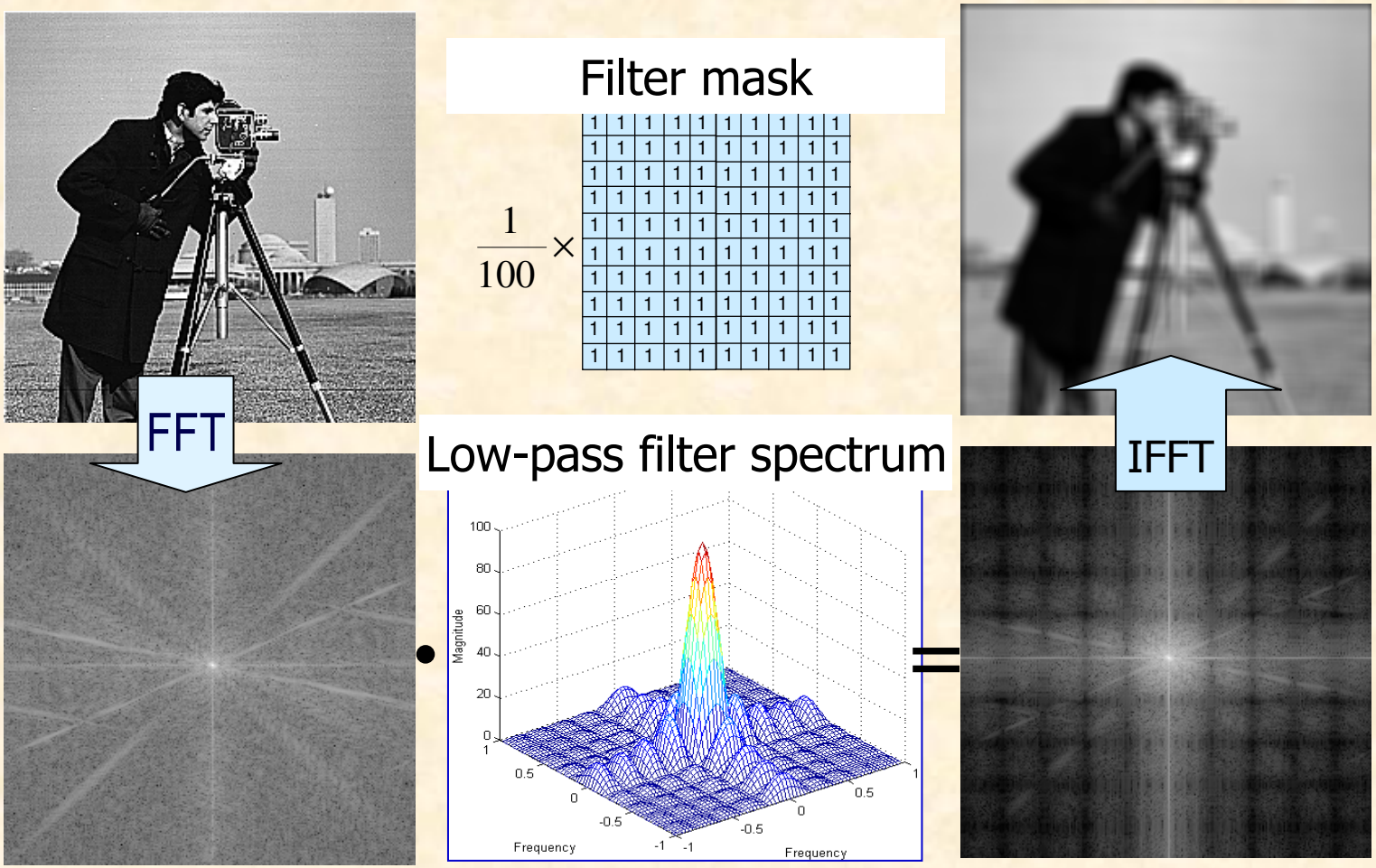
High-pass filters

Blurred image

Sharpened image

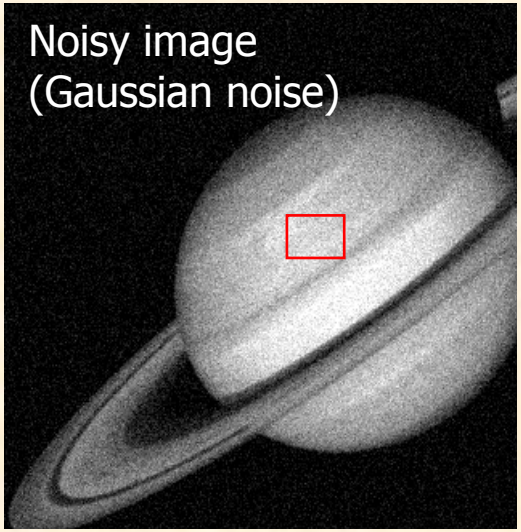
```
%MATLAB
out_image = filter2(filter_mask, in_image);
```

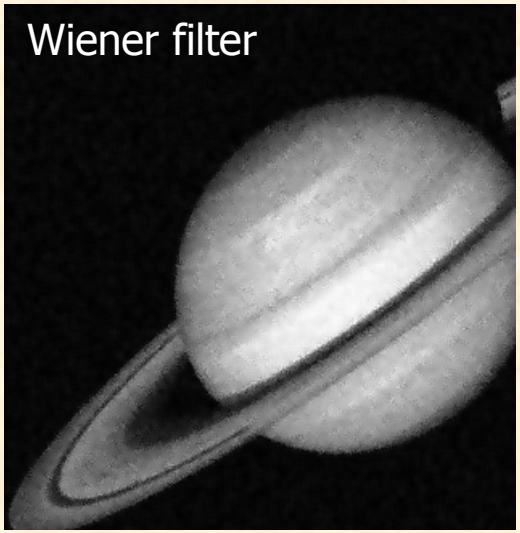
Filtering in spectrum domain

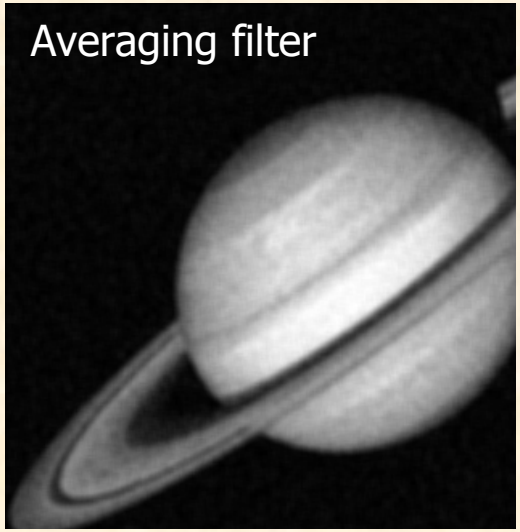


P. Strumiłło, M. Strzelecki

The Wiener adaptive filter







$$g(x,y) = \mu + \frac{\sigma^2 - n^2}{\sigma^2}\left(f(x,y) - \mu\right)$$

For: $n \cong 0,\ \ g(x,y) \cong f(x,y)$

$n \cong \sigma,\ \ g(x,y) \cong \mu$

where:

n^2 – noise variance

μ – mean of image pixels in a window

σ^2 – image variance

P. Strumiłło, M. Strzelecki

Nonlinear filters

The filtered image is defined by a non-linear function of the source image

Can we compute spectral characteristics for nonlinear filters?

NO

Because transfer characteristics of nonlinear filters depend on image content itself!

Median filter (order statistic filter)

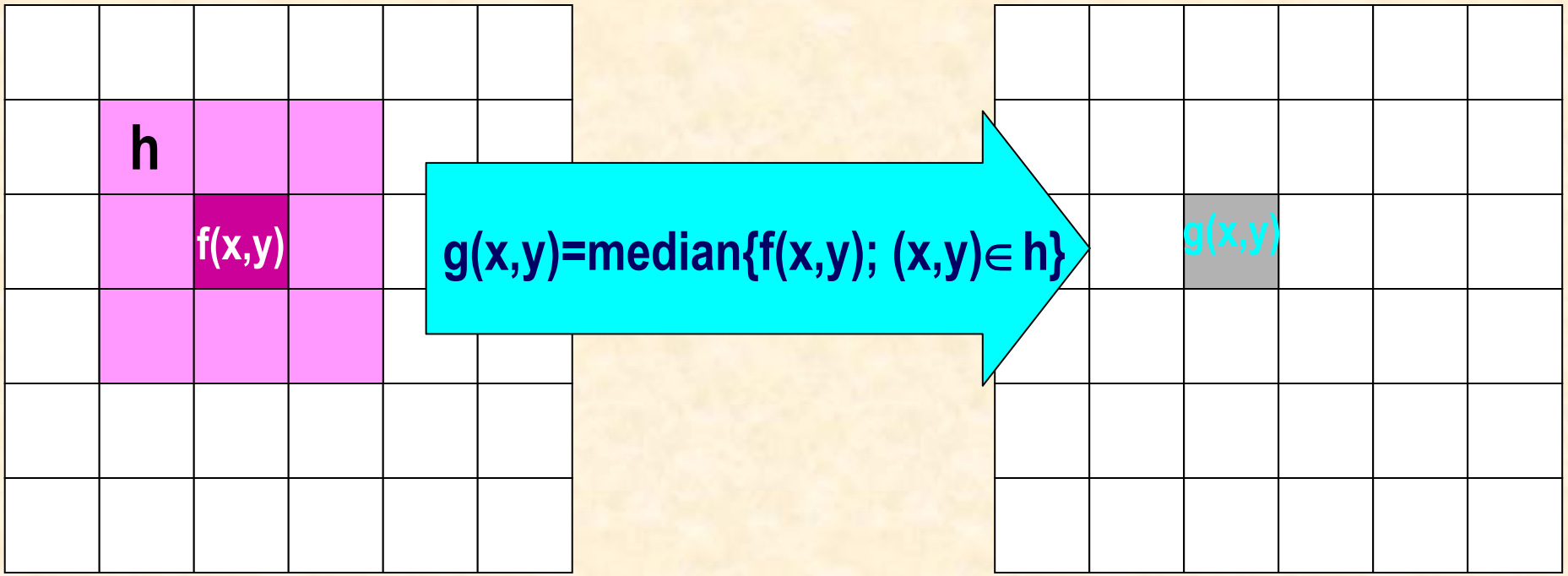
The median m of a set of values (e.g. image pixels in the filtering mask) is such that half the elements in the set are less than m and other half are grater than m.

$$x(n)=\{1, 5, -7, 101, -25, 3, 0, 11, 7\}$$

Sorted sequence of elements:

$$x_s(n)=\{-25, -7, 0, 1, \mathbf{3}, 5, 7, 11, 101\}$$

Median filtering the image

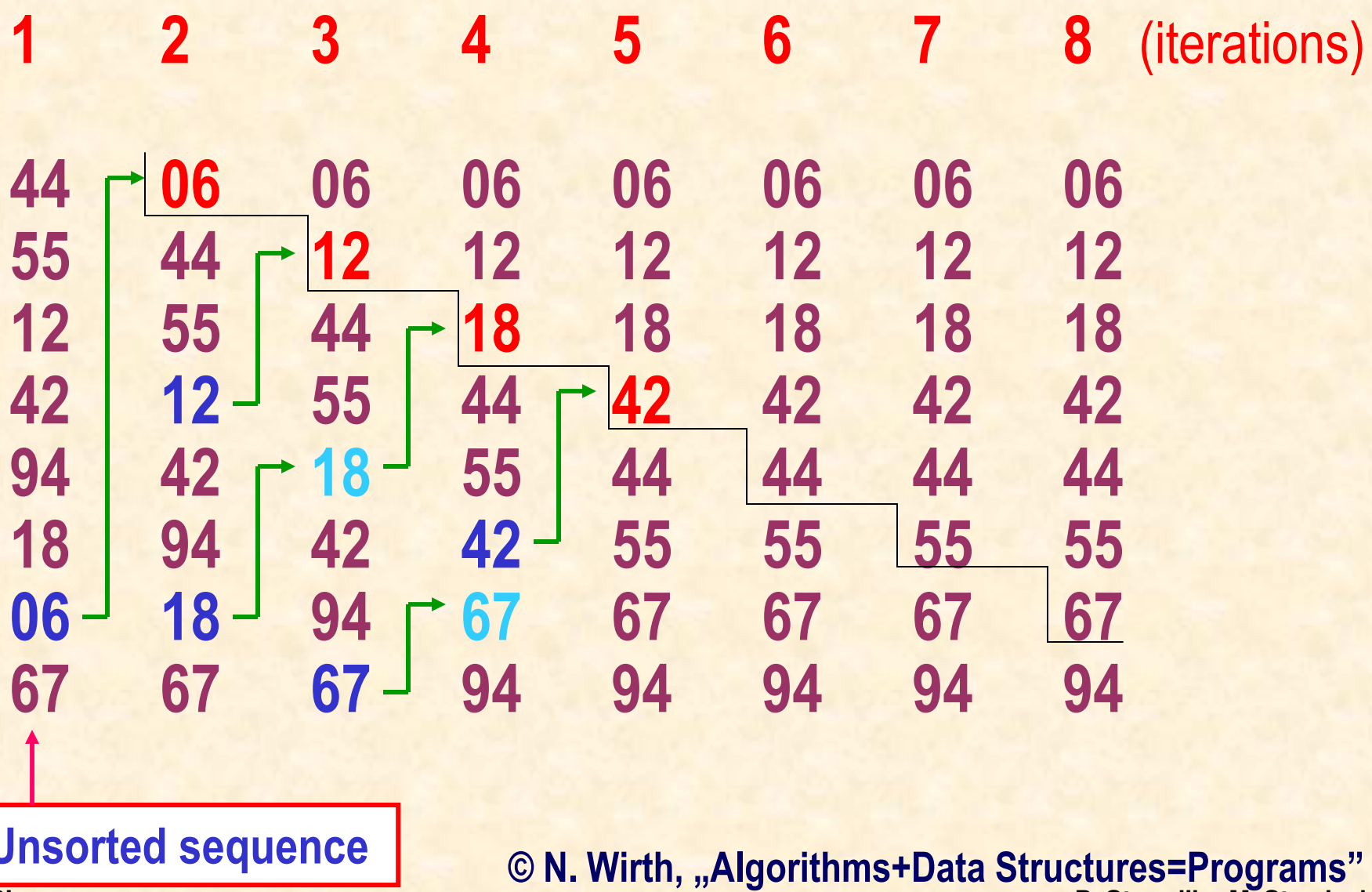


source image f

output image g

P. Strumiłło, M. Strzelecki

Bubble-sort algorithm

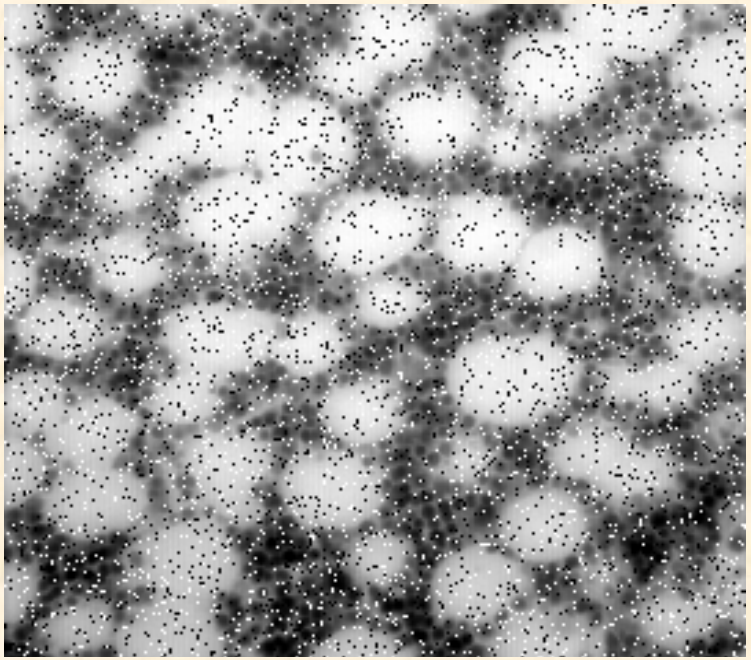


© N. Wirth, „Algorithms+Data Structures=Programs”

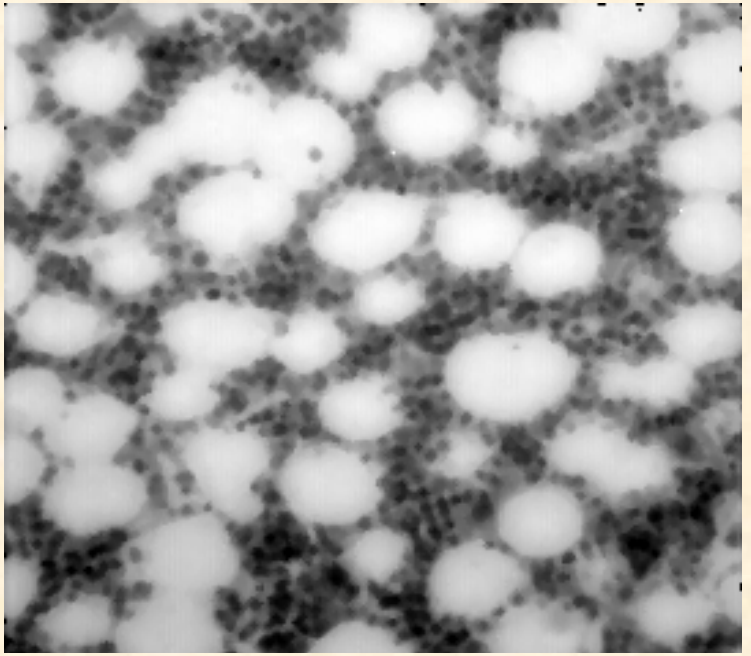
Bubble-sort program

```
a[k], k=1..N – unsorted sequence
for i:=2 to N do
begin
   for j:=N downto i do
   if a[j-1]>a[j] then
      begin
      x=a[j-1]; a[j-1]:=a[j]; a[j]:=x;
      end;
end;
```

Demo – median filter



Source image distorted by „salt and pepper noise”



Enhanced image using the median filter (3x3)”

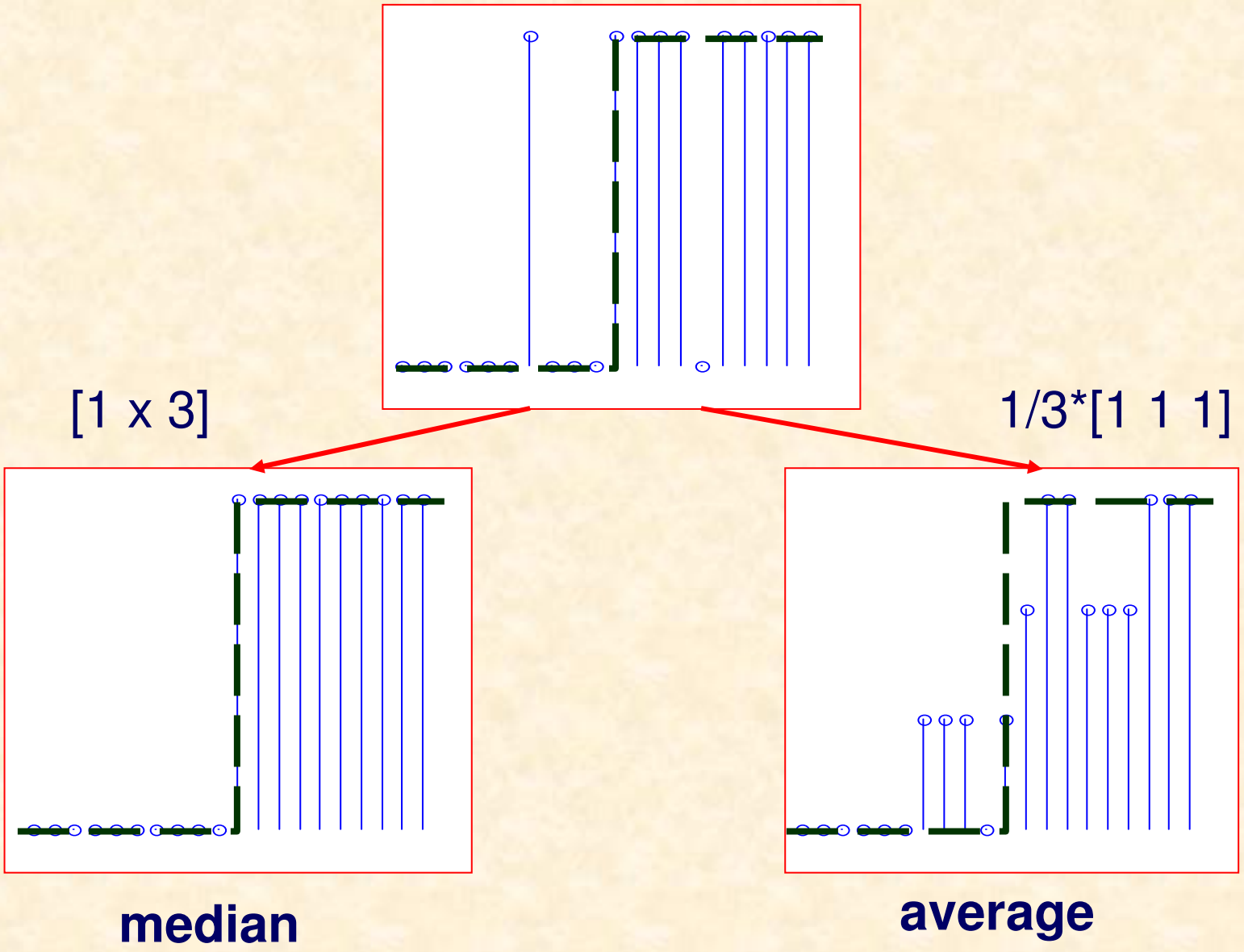
```
%MATLAB
out_image = medfilt2(in_image, [m n]);
```

Median filter

Median filter:

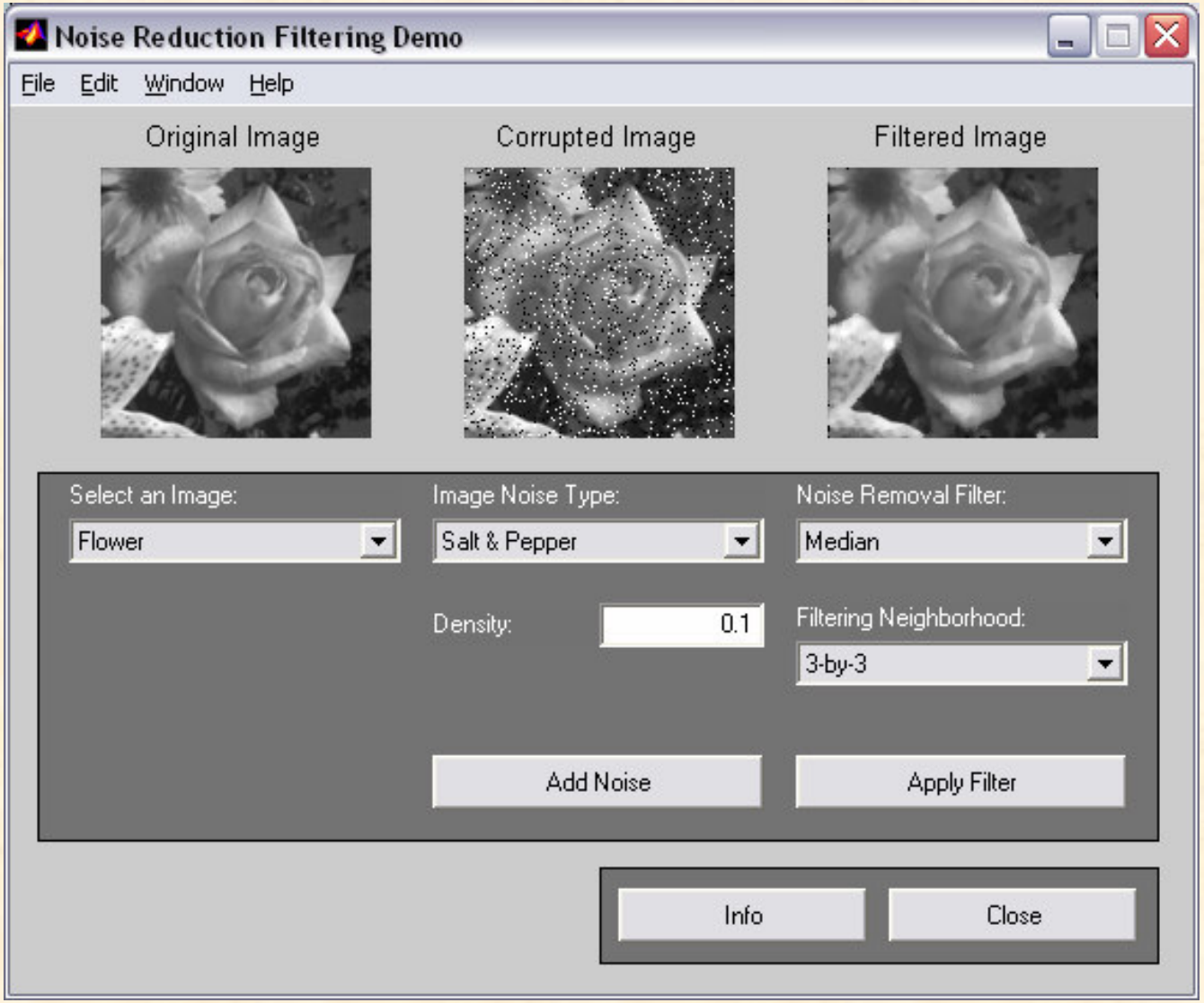
1. Excellent in reducing impulsive noise (of size smaller than half size of the filtering mask)
2. Keeps sharpness of image edges (as opposed to linear smoothing filters)
3. Values of the output image are equal or smaller than the values of the input image (no rescaling required)
4. Large computing cost involved

Median filter

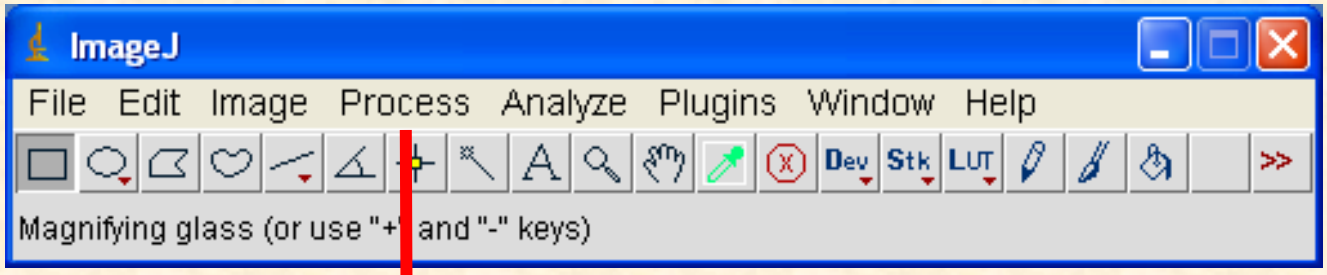


P. Strumiłło, M. Strzelecki

MATLAB Demo – median filter



ImageJ



- Smooth
- Sharpen
- Filters
  - Convolve
  - Gaussian Blur
  - Median
  - Mean
  - Unsharp mask