TÉCNICO LISBOA

TÉCNICO LISBOA

Principles of Biosignals and Biomedical Imaging

3rd year, P_3 (ECTS: 3.0), LEBiom

2022/2023

João Sanches
jmrs@tecnico.ulisboa.pt
Department of Bioengineering
Instituto Superior Técnico
University of Lisbon

TÉCNICO LISBOA

Intensity Transformation

Signal-to-Noise ratio

$$SNR = 20log\left(\frac{S}{N}\right)$$

Let X and Y the original signal (1D, image or volume) and Y the corresponding reconstruction (denoised, deblurred).

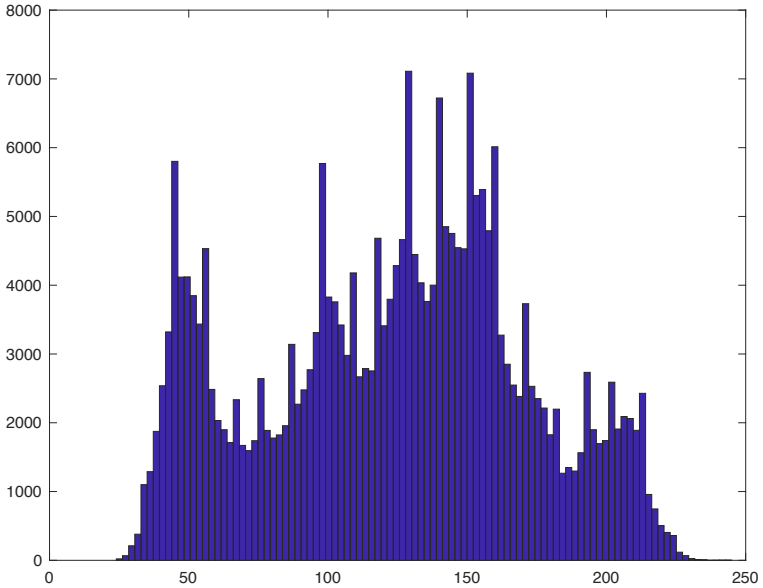
$$SNR_{dB} = 20log\left(\frac{S}{N}\right)$$

where S and N are the power of the signal and of the noise respectively.

$$S = \|X\|^2 = \sum_{i,j} x_{i,j}^2$$

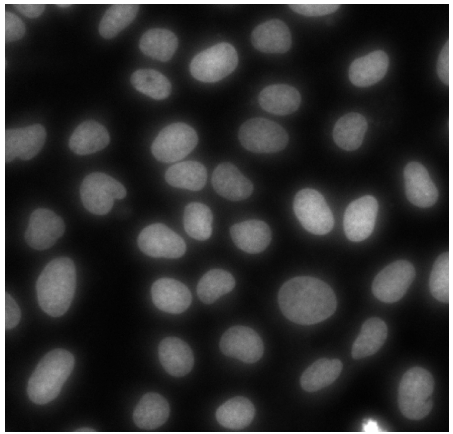
$$\text{N} = \|X - Y\|^2$$

Image Histogram

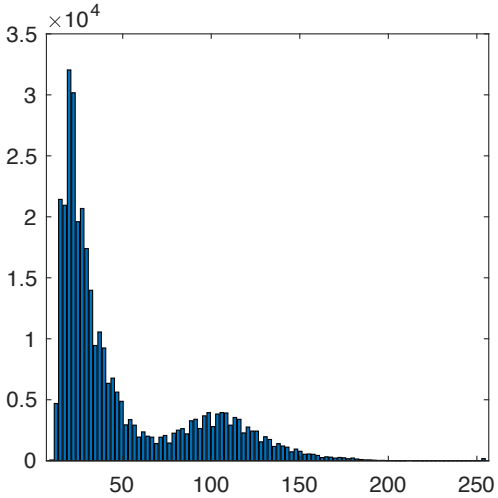


$$h(k) = \#(x_{i,j} = k)$$

Image Histogram

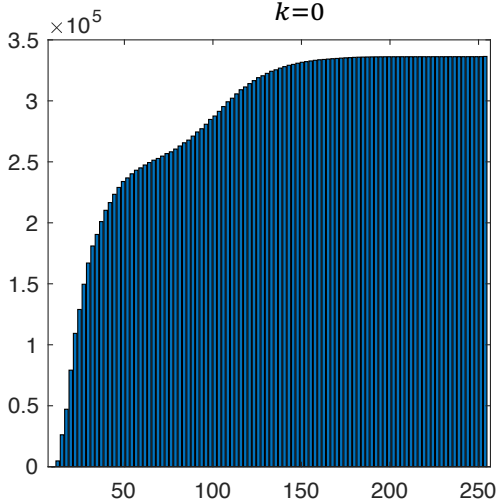


$h(n)$



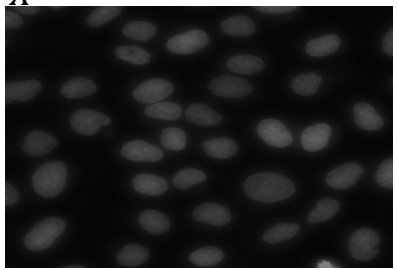
Cumulative Histogram

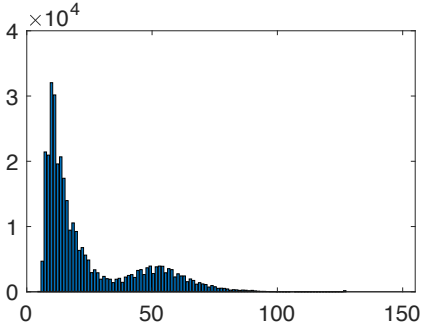
$$c(n) = \sum_{k=0}^{n} h(k)$$



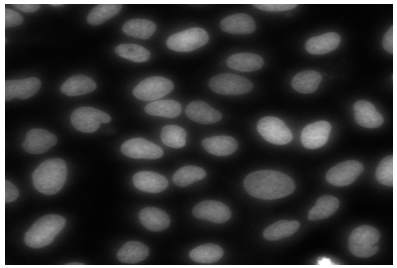
Intensity normalization

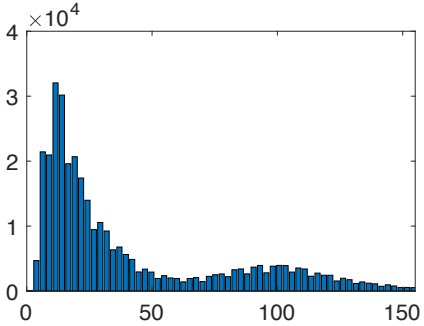
X





Y

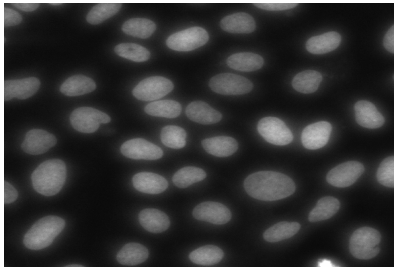


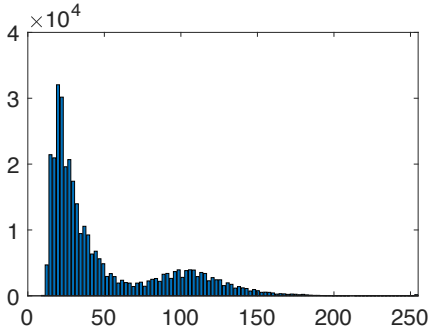


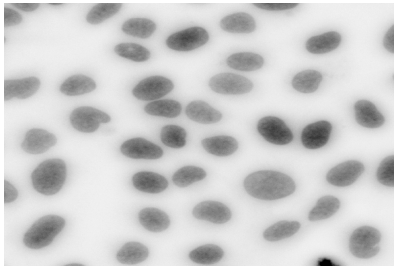
$$X = \{x_{i,j}\}$$
$$Y = \{y_{i,j}\}$$

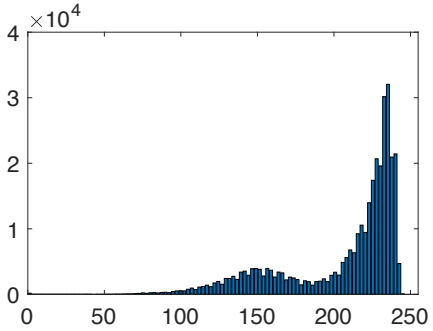
$$y_{i,j} = 255\frac{x_{i,j} - \min(x_{i,j})}{max(x_{i,j}) - \min(x_{i,j})} = ax_{i,j} + b$$

Intensity inversion





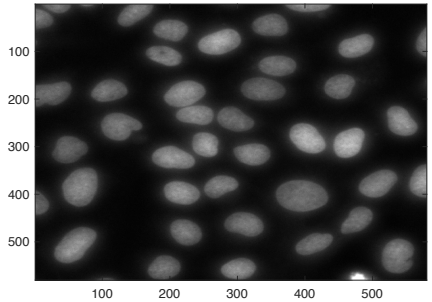


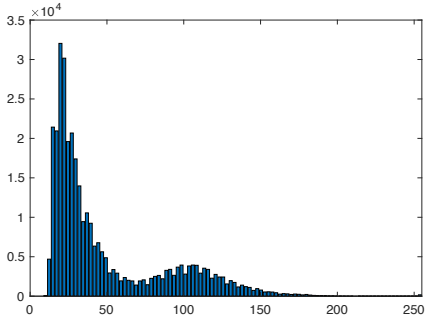


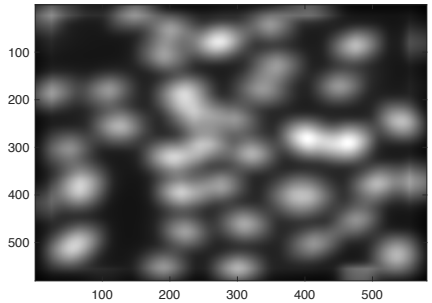
$X = \{x_{i,j}\}$
$Y = \{y_{i,j}\}$

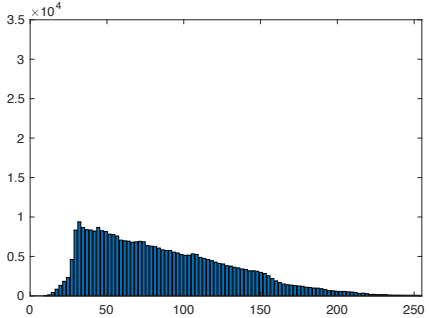
$y_{i,j} = 255 - x_{i,j}$

Image filtering (smoothing)










```
%Read image
img = 255*double(imread('CellsImage.png'))/65535;

%image filtering with a moving average rectangular window
N=50;
imgOut=conv2(img, ones(N)/N^2,'same');

%Histogram computation
[h,x]   = hist(img(:),100);
[hO]    = hist(imgOut(:),100);

 %Results
figure(1); colormap(gray(256));

subplot(2,2,1); imagesc(img);
subplot(2,2,2); bar(x, h); xlim([0,255]);ylim([0 35000]);

subplot(2,2,3); imagesc(imgOut);
subplot(2,2,4); bar(x, hO); xlim([0,255]);ylim([0 35000]);
```

Histogram equalization

- Let $x_{i,j} \in \{0,1,2,\dots,L-1\}$ be the intensity of the $(i,j)^{th}$pixel where L is the number of different intensities

- Let $0 \leq p(k) = \frac{\#(x_{i,j}=k)}{N} \leq 1$where N is the total number of pixels

- $p(k)$ is the probability of a pixel $x_{i,j}$ has intensity k

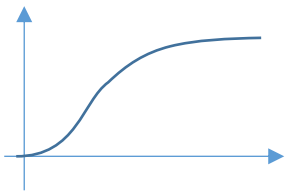
- $p(k) = h(k)/N$ where $h(k) = \#(x_{i,j} = k)$ is the number of pixels with intensity k

- $\sum_{k=0}^{L-1} p(k) = \frac{1}{N}\sum_{k=0}^{L-1} h(k) = \frac{1}{N}N = 1$

Histogram equalization

- Let x be a random variable: $x \sim p_X(x)$

- Let $y = f(x)$ where $f(x)$ is a monotonic growing function



- $p_Y(y) = \frac{dx(y)}{dy}\ p_X\big(x(y)\big) = \frac{df^{-1}(y)}{dy}\ p_X(f^{-1}(y))$

Histogram equalization

- Let $y = f(x) = \int_0^x p_X(\tau)d\tau = P_X(x)$
  the cumulative distribution function of x **then**

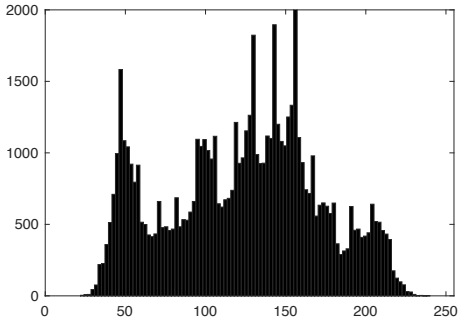
$$p_Y(y) = 1$$

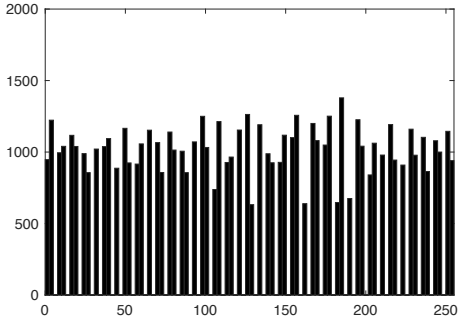
Proof

Let $f(x) = \int_0^x p_X(\tau)d\tau \Rightarrow \frac{df(x)}{dx} = p_X(x)$

$$p_Y(y) = p_X(x)\frac{dx}{dy} = \frac{df(x)}{dx}\cdot\frac{dx}{dy} = \frac{df(x)}{dy} = \frac{d[f(f^{-1}(y)]}{dy} = \frac{dy}{dy} = 1$$

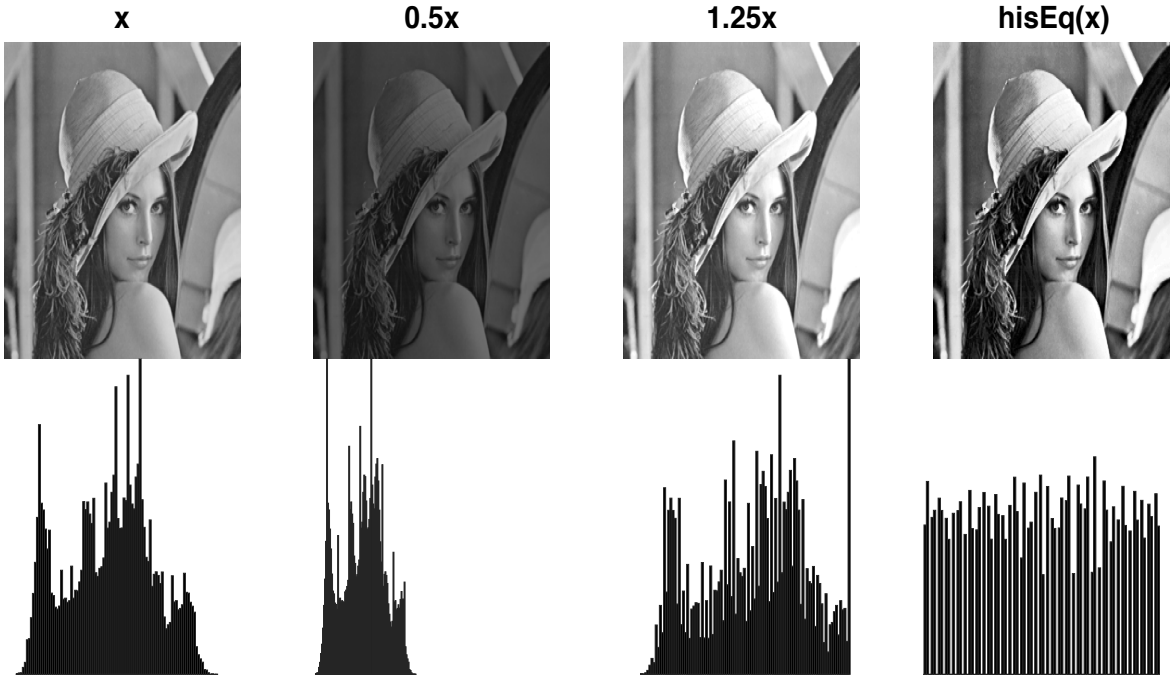
Histogram equalization






```
%% Histogram Equalization

%Read image
x=imread('len_gray.jpg');

%Resize image
y=imresize(x,0.5);

%Histogram equalization
z=histeq(y);

%Display results
figure(1); colormap(gray(256));
subplot(2,2,1); image(y); axis off;
subplot(2,2,2);hist(double(y(:)),100);
   xlim([0,255]); ylim([0,2000]);

subplot(2,2,3); image(z); axis off;
subplot(2,2,4);hist(double(z(:)),100);
   xlim([0,255]); ylim([0,2000]);
```

Histogram




```
%% Histogram
x=imread('len_gray.jpg');
y=imresize(x,0.5);

figure(2); colormap(gray(256));

subplot(2,4,1); image(y);
title('x', 'fontsize', 18);axis off;
subplot(2,4,5); hist(double(y(:)),100);
axis off; xlim([0,255]); ylim([0,2000]);

subplot(2,4,2); image(0.5*y);
title('0.5x', 'fontsize', 18);axis off;
subplot(2,4,6);
hist(double(0.5*y(:)),100);
axis off; xlim([0,255]); ylim([0,2000]);

subplot(2,4,3); image(1.25*y);
title('1.25x', 'fontsize', 18);  axis off;
subplot(2,4,7);
hist(double(1.25*y(:)),100);
axis off; xlim([0,255]); ylim([0,2000]);

subplot(2,4,4); z=histeq(y); image(z);
title('hisEq(x)', 'fontsize', 18);  axis
off;
subplot(2,4,8); hist(double(z(:)),100);
axis off; xlim([0,255]); ylim([0,2000]);
```

TÉCNICO LISBOA

Interval

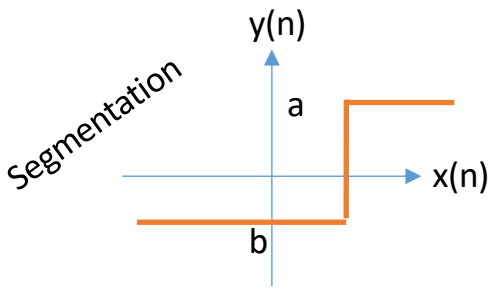
TÉCNICO LISBOA

Non-linear filtering

Non-Linear filters

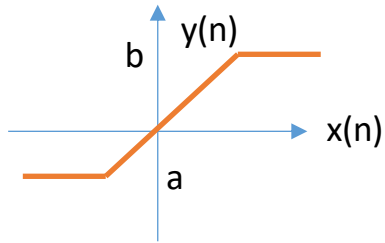
- Binarization

$$y(n) = \begin{cases} a & if \quad x(n) \geq th \\ b & otherwise \end{cases}$$



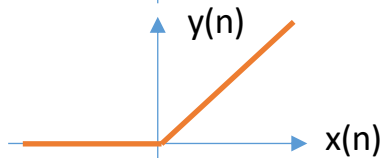
- Saturation

$$y(n) = \begin{cases} x(n) & if \quad a < x(n) < b \\ a & if \; x(n) < a \\ b & if \; x(n) > b \end{cases}$$



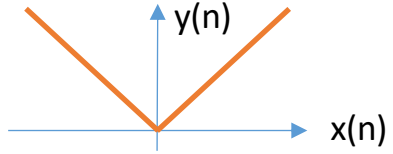
- Half Rectifier

$$y(n) = \begin{cases} x(n) & if \quad x(n) > 0 \\ 0 & otherwise \end{cases}$$

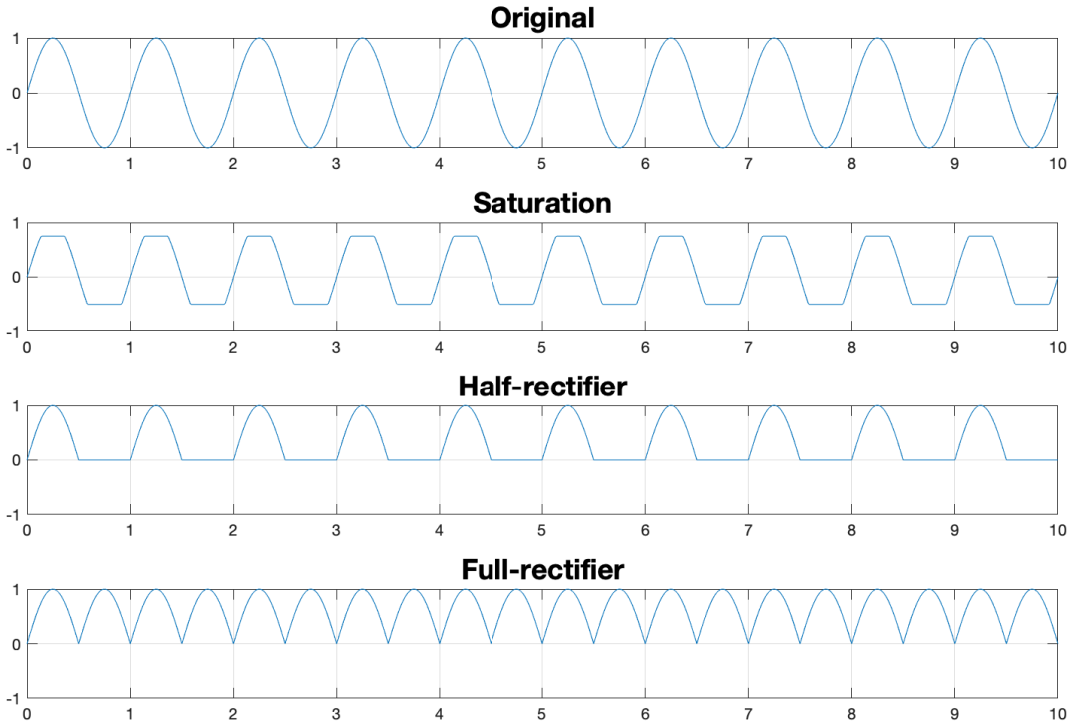


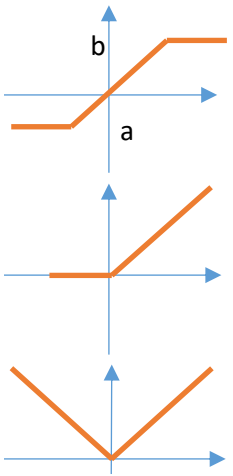
- Full Rectifier

$$y(n) = abs\big(x(n)\big) = \begin{cases} x(n) & if \quad x(n) > 0 \\ -x(n) & otherwise \end{cases}$$



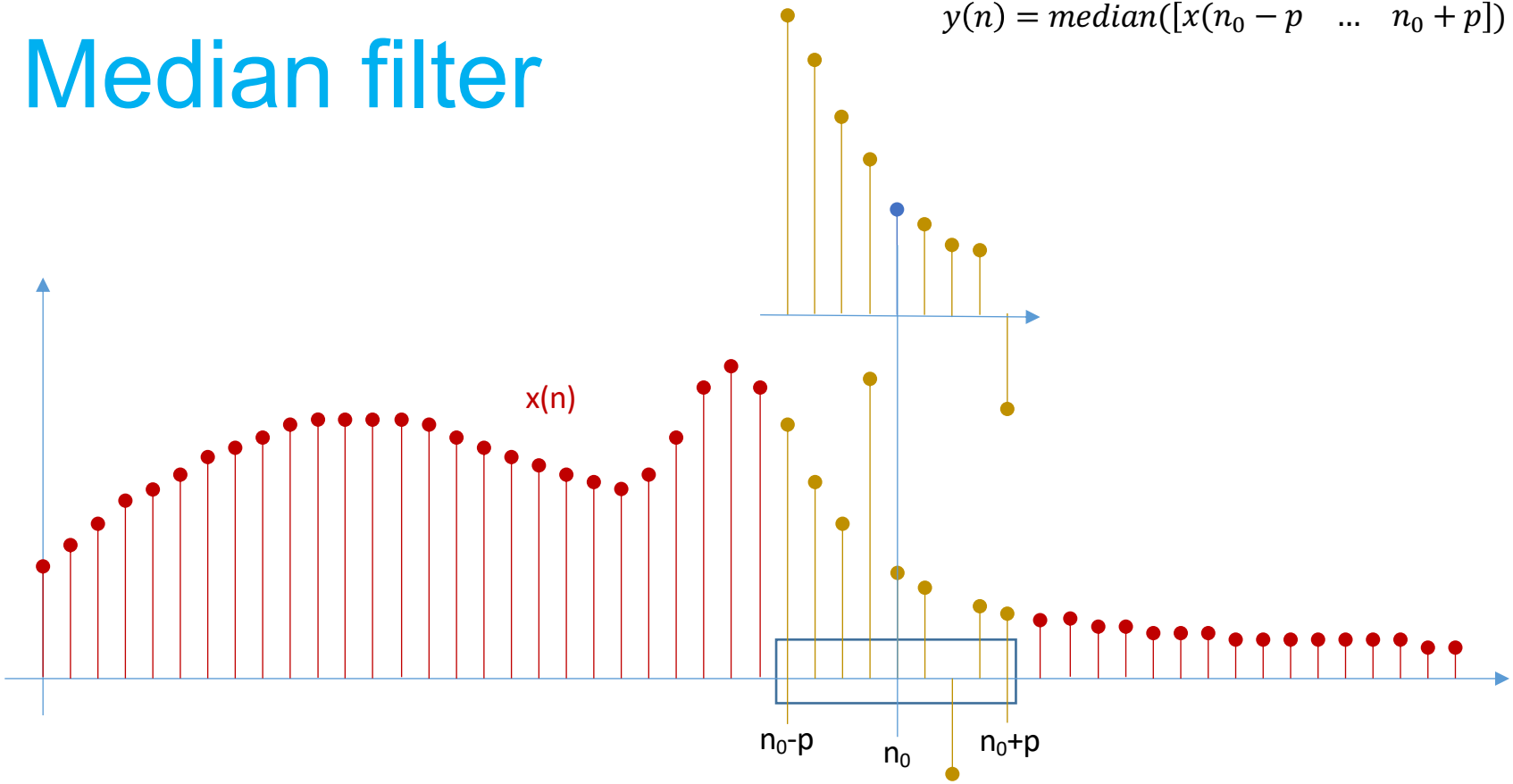
Non-Linear filters



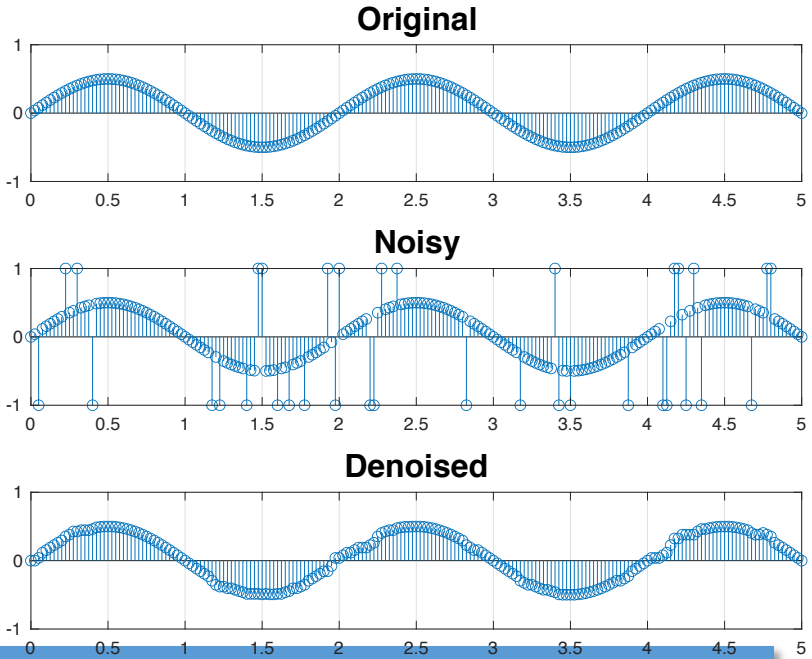


Median filter

$$y(n) = median([x(n_0 - p \quad \ldots \quad n_0 + p])$$



Non-Linear filters - Median filter



SNR=-3.3dB

SNR=55.26 dB

The output of the filter is the median of the set with N elements containing the current input and the N-1 neighbors

```
%Median Filter

%Initialization
T=5;          %Simulation time
fs=40;        %Samling frequency
Ts=1/fs;      %Sampling period
t=(0:Ts:T)';  %Simulation times

f=0.5;        %Harmonic frequency

%Signals generation
   x=0.5*sin(2*pi*f*t);    %Original harmonic
%Original signal corrupted with impuse noise
   z=zeros(size(x));
   for I=1:length(z)
   a=rand;         %a~~Unif(0,1)
   if a>0.8        % noise intensity
      z(I)=2*((rand>0.5)-0.5);
   else
      z(I)=x(I);
   end
   end
%Cleaning with a median filter
y = medfilt1(z,3,'zeropad');

figure(1);
subplot(3,1,1); stem(t,x); grid; ylim([-1 1]); title('Original',  'Fontsize',18);
subplot(3,1,2); stem(t,z); grid; ylim([-1 1]); title('Noisy',     'Fontsize',18);
subplot(3,1,3); stem(t,y); grid; ylim([-1 1]); title('Denoised',  'Fontsize',18);
```

Median filter

Original

Noisy: SNR(DB)=14.82

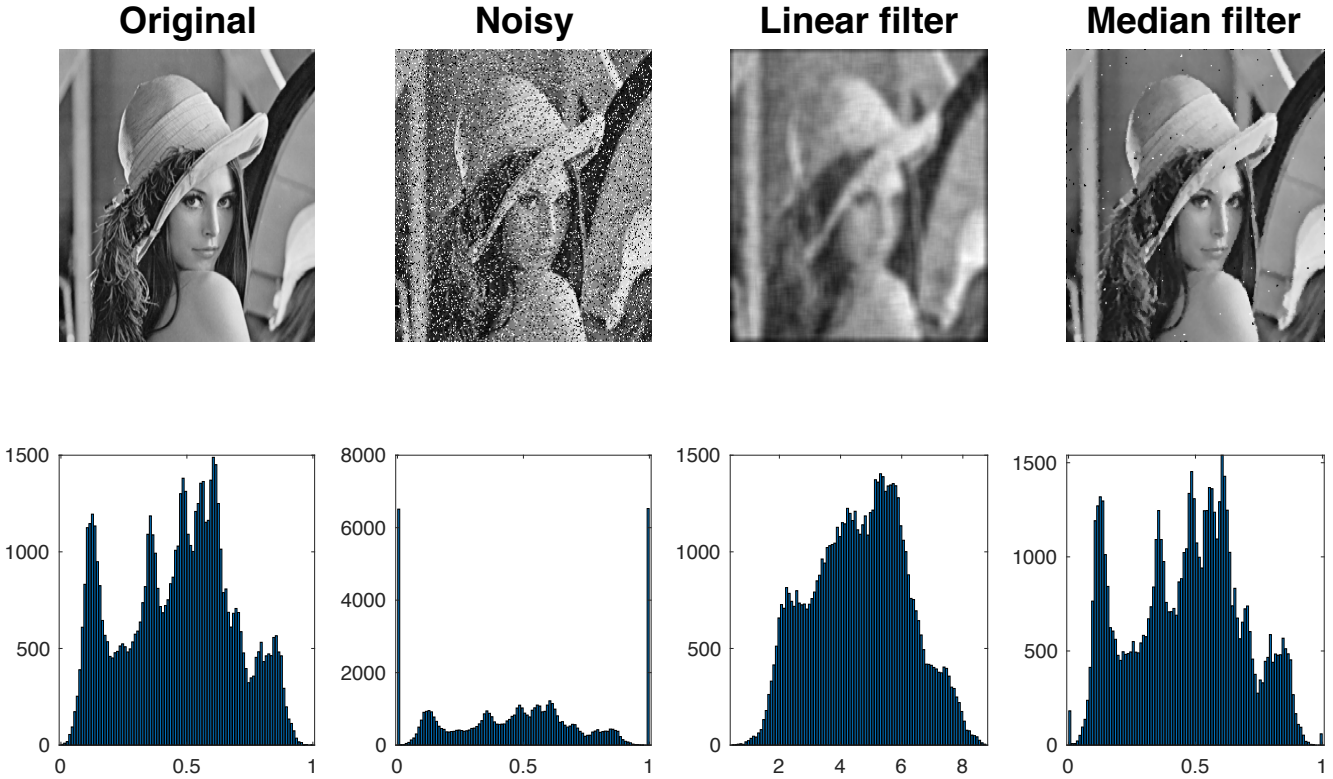
Linear filter: SNR(DB)=-42.82

Median filter: SNR(DB)=47.83

```
%% ------ Imagem
% Original
x=double(imread('len_gray.jpg'));
x=(x-min(x(:)))/(max(x(:))-min(x(:)));
% Noisy with salt & papper
y=imnoise(x,'salt & pepper', 0.2);
% Filtered with median filter
z=medfilt2(y);
% Linear filtering
w=conv2(y, ones(10)/10,'same');
```

```
figure(1);
colormap(gray(256));
subplot(2,2,1); imagesc(x);
axis off; title('Original', 'fontsize', 18);
subplot(2,2,2); imagesc(y);
axis off; title('Noisy', 'fontsize', 18);
subplot(2,2,3); imagesc(w);
axis off; title('Linear filter', 'fontsize', 18);
subplot(2,2,4); imagesc(z);
axis off; title('Median filter', 'fontsize', 18);
```

Median filter



TÉCNICO LISBOA

Exemplo

Non-linear filtering

```
%%  Non-Linear Filtering
% Original
img=double(imread('len_gray.jpg'));
x=imresize(img,0.5);

thrs=128;
y=(x>thrs);

th1=50;     th2=150;
w=(x>th1).*(x<th2).*x;

thrs=50;
z=conv2(x,[0 -1 0; -1 0 1; 0 1 0],'same');
zo=(z<-thrs)+(z>thrs);

figure(1); colormap(gray(256));
subplot(1,4,1); imagesc(x);  axis off;
subplot(1,4,2); imagesc(y);  axis off;
subplot(1,4,3); imagesc(w);  axis off;
subplot(1,4,4); imagesc(zo); axis off;
```

TÉCNICO LISBOA