Technical University of Cluj - Napoca
Computer Science Department

Sisteme de viziune in robotica

An2, Master Robotica

Cuprins / Contents

Grayscale image processing

1. Proprietati statistice ale imaginilor grayscale si aplicatii. Imbunatatirea calitatii imaginilor (*Statistical properties of grayscale images and applications. Image Enhancement*)
2. Filtrarea imaginilor / filtre spatiale (*Image filtering*)
3. Modelarea si eliminarea zgomotelor (*Noise modeling & removal*)
4. Detectia muchiilor / metoda de segmentare bazata pe discontinuitati (*Edge detection & segmentation*)
5. Detectia colturilor (*Corner detection*)

Statistical properties of grayscale images and applications. Image Enhancement

Proprietati statistice ale imaginilor grayscale. Imbunatatirea calitatii imaginilor

Trasaturi statistice / *Statistical features*

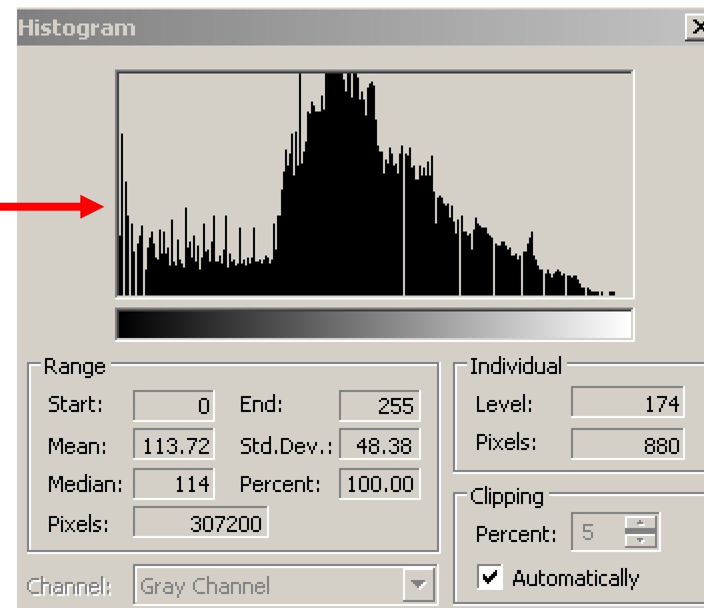
Statsitical features$\Rightarrow$ global features (computed on the whole image or on ROIs)

Histograma intensitatilor (*Image histogram*)

Gray level: g $\in$ [0 … L], *L* – max. level (8bits/pixel images: L= 0 .. 255)

$$h(g)=N_g$$

N_g – no. of pixels in the image / ROI having the brightness level *g*



Probability distribution function (of the gray-levels)

(functia distributiei de probabilitate a nivelelor de gri)

P(g) := probability for the brightness in a region <= g:

$0 <= P(g) <= 1$

P(g) – monotonically increasing $\Rightarrow dP/dg >= 0$

Probability density function (of the gray-levels): *p(g)*

(functia densitatii de probabilitate a nivelelor de gri)

$p(g)\Delta g$:= probability for the brightness in a region being between $g \ldots g+\Delta g$

$$p(g)\cdot\Delta g = \left(\frac{dP(g)}{dg}\right)\Delta g$$

p(g) – normalized histogram :

$$p(g) = \frac{h(g)}{M}, \quad with: \begin{cases} p(g) >= 0 \\ \int_{-\infty}^{\infty} p(g)dg = 1, \quad \sum_{g=0}^{L} \frac{h(g)}{M} = \frac{M}{M} = 1 \end{cases}$$

M = image_height x image_width

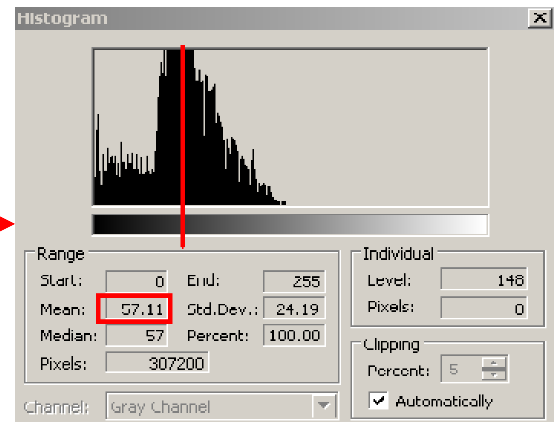
Mean (media intensitatilor)

⇒ Measure of the average brightness of the image / ROI

$$\bar{g} = \mu = \int g \cdot p(g)dg = \sum_{g=0}^{L} g \cdot p(g) = \frac{1}{M}\sum_{g=0}^{L} g \cdot h(g) = \frac{1}{M}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} I(i,j)$$

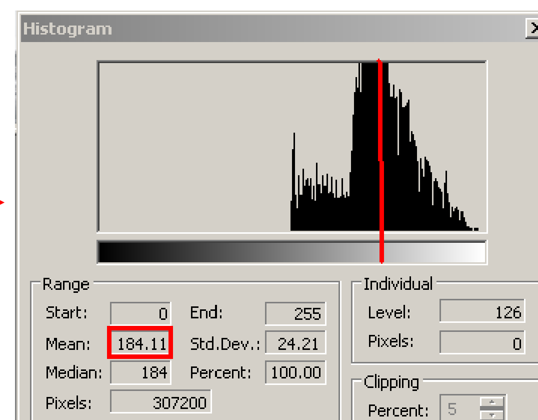
Image "intunecata"

(dark image)



Imagine "luminoasa"

(bright image)



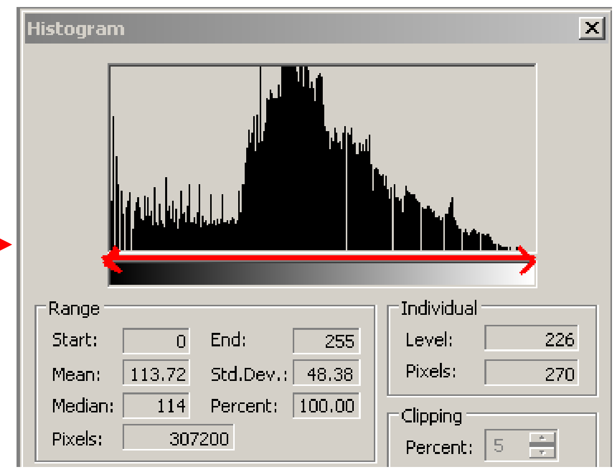
Standard deviation (deviatia standard)

⇒ Measure of the average contrast of the image / ROI

$$\sigma = \sqrt{\sum_{g=0}^{L}(g-\mu)^2 \cdot p(g)} = \sqrt{\frac{1}{M}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left(I(i,j)-\mu\right)^2}$$

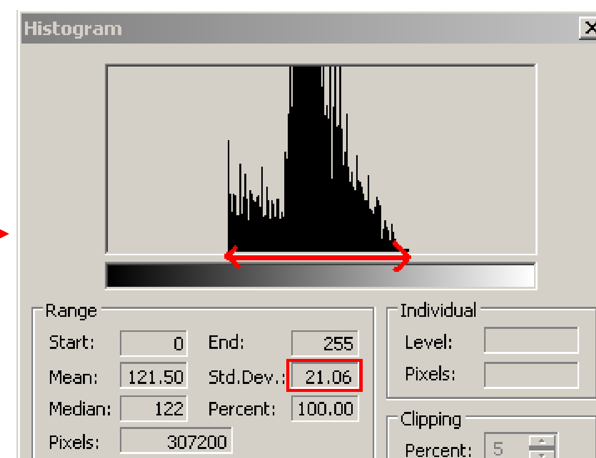
Contrast ridicat

(high contrast)



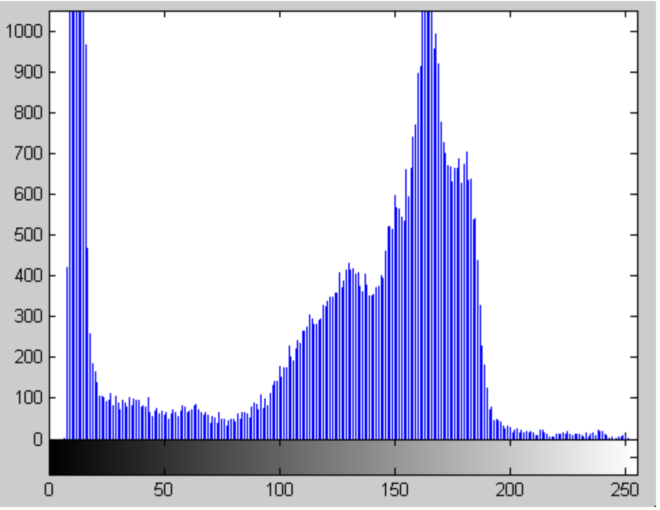
Contrast scazut

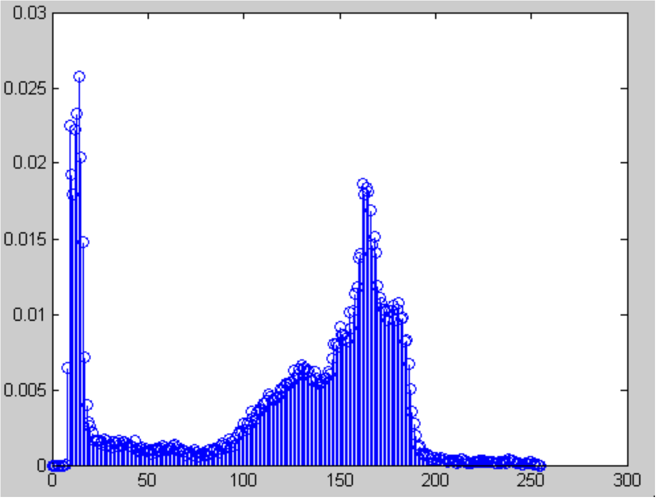
(low contrast)



Exemple: computing statistical features

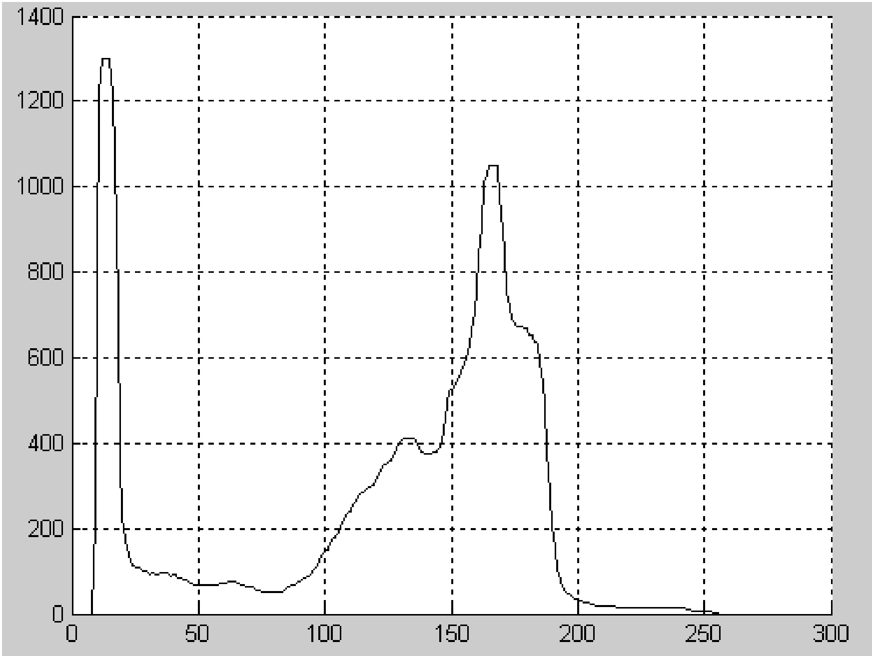
```
%Compute statistical properties: histogram, FDP,
mean, standard deviation

%read bitmap image from file cameraman.bmp
I=imread('cameraman.bmp','BMP');
ColorDepth=256;

[height,width]=size(I);
M=height*width;

%compute histogrm
[h,x] = imhist(I,ColorDepth);
%display histogram
figure(1); imhist(I,ColorDepth)

%compute FDP
p = h / M;
%display  using stem function
figure(2);stem(x,p);

%display histogram using plot function
figure(3); hold on; grid on; plot(h,'k-');hold off;

%histogram filtering with a median filter
h=medfilt1(h,10);
%display filtered histogram using plot function
figure(4); hold on; grid on; plot(h,'k-');hold off;

%compute brightness mean
medie = mean2(I)
%compute brightness standard deviation
std_dev = std2(I)
```





Exemple: computing statistical features

Histogram displayed using plot function



Filtered histogram displayed using plot function

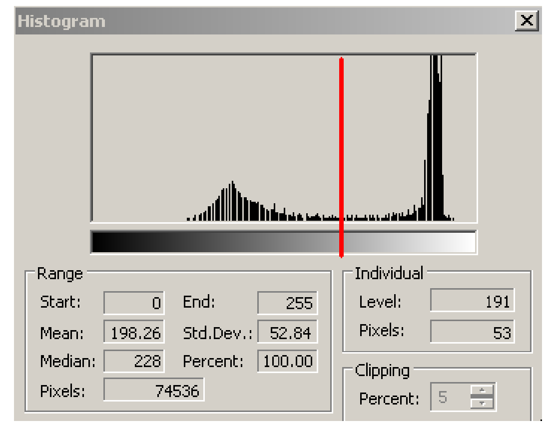
mean = 118.7245

std_dev = 62.3417

Application: grayscale image segmentation

Grayscale image segmentation by adaptive thresholding

- Automatic computation of the threshold (T)
- Can be applied on images with bi-modal histogram



The algorithm

1. Take an initial value for T:

   $T_0 = \mu$ (object area = background area)

   $T_0 = (g_{MAX} + g_{MIN})/2$

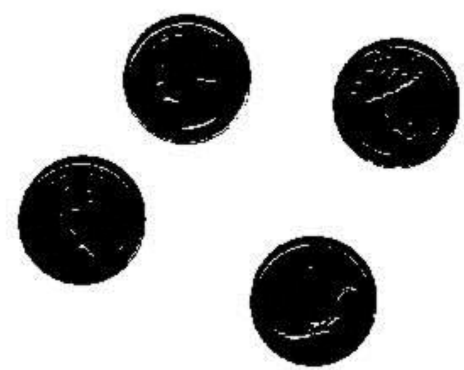
2. *Step k*: segment the image after T_{k-1} by dividing the image pixels in 2 groups:

   (Foreground) $G1: I[i,j] < T_{k-1} \Rightarrow \mu_{G1}$

   (Background) $G2: I[i,j] > T_{k-1} \Rightarrow \mu_{G2}$

3. $T_k = (\mu_{G1} + \mu_{G2})/2$

4. Repeat 2-3 until $T_k - T_{k-1} < \varepsilon$



Efficient implementation $\Rightarrow$ first the image histogram is computed then all computations (μ_{Gi}) will be done on the histogram !!!

Exemple: adaptive thresholding

```
function T=hist_threshold(I);
% Compute binary treshold using the "The global adaptive threshold method"

[height,width]=size(I);
M=height*width;
ColorDepth=256;
[h,x] = imhist(I,ColorDepth);

T=(max(max(double(I)))+min(min(double(I))))/2;
Te=0.5;
dT=256;
i=0;

while dT > Te,
To=round(T);
    m1=0; m2=0;
    nr_pix=0;
    for z=1:To,
        nr_pix=nr_pix+h(z);
        m1=m1+z*h(z);
    end
    m1=m1/nr_pix;
    nr_pix=0;
    for z=To+1:ColorDepth,
        nr_pix=nr_pix+h(z);
        m2=m2+z*h(z);
    end
    m2=m2/nr_pix;
    T=(m1+m2)/2;
    dT=abs(To-T);
end
T=round(T);
```

```
%Usage
I=imread('eight.bmp','BMP');
ColorDepth=256;
T=hist_threshold(I);
T_norm = T/ ColorDepth
Ibw=im2bw(I, T_norm);
Ibw=~Ibw;
figure; imshow(Ibw);
```

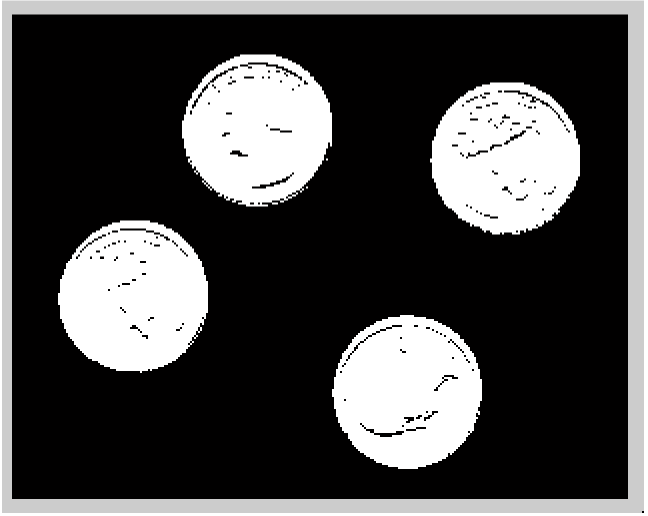
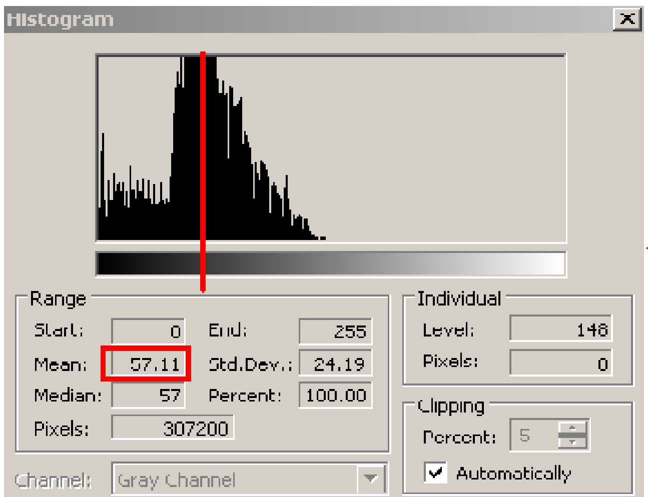


Image enhancement / Imbunatatirea calitatii imaginilor

Histogram slide / deplasarea histogramei

$$Slide(I[i,j]) = I[i,j] + offset$$

offset > 0 ⇒ “brighter” image
offset < 0 ⇒ “darker” image



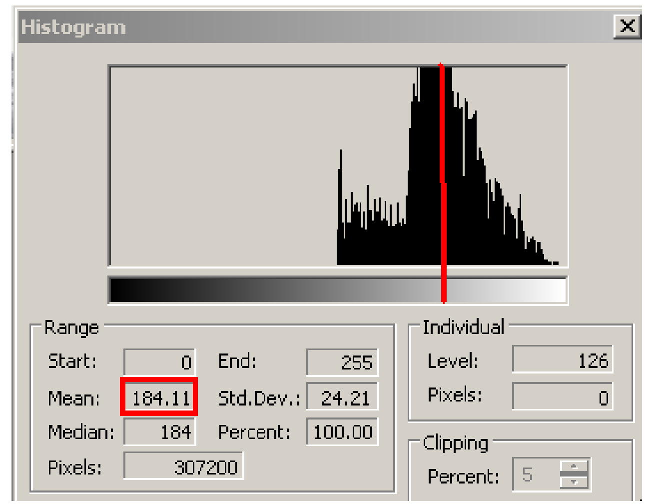
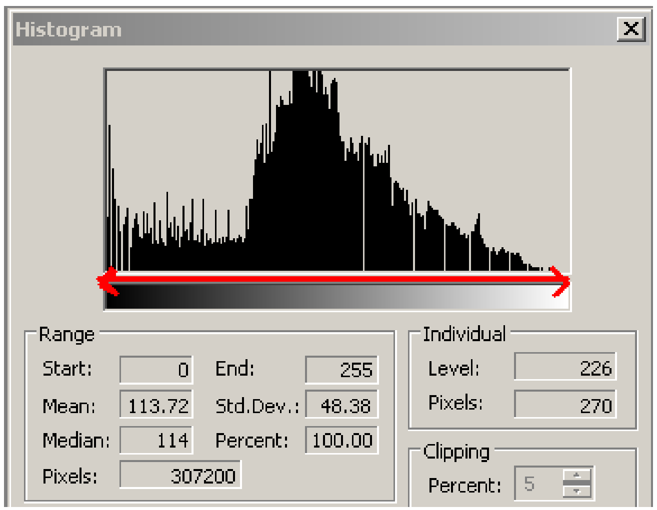


Image enhancement / Imbunatatirea calitatii imaginilor

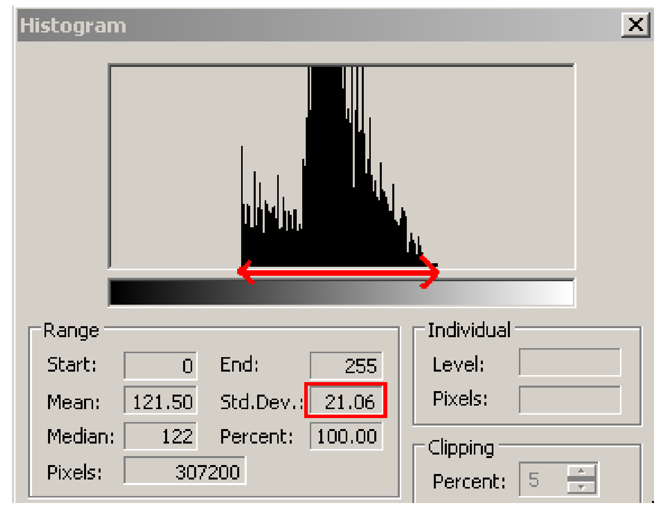
Histogram stretch/shrink (latire/ingustare a histogramei)

$$\textit{Strecth/Shrink}(I[i,j]) = Final_{MIN} + (Final_{MAX} - Final_{MIN})*(I[i,j] - g_{MIN}) / (g_{MAX} - g_{MIN})$$



shrink →

← stretch



shrink →

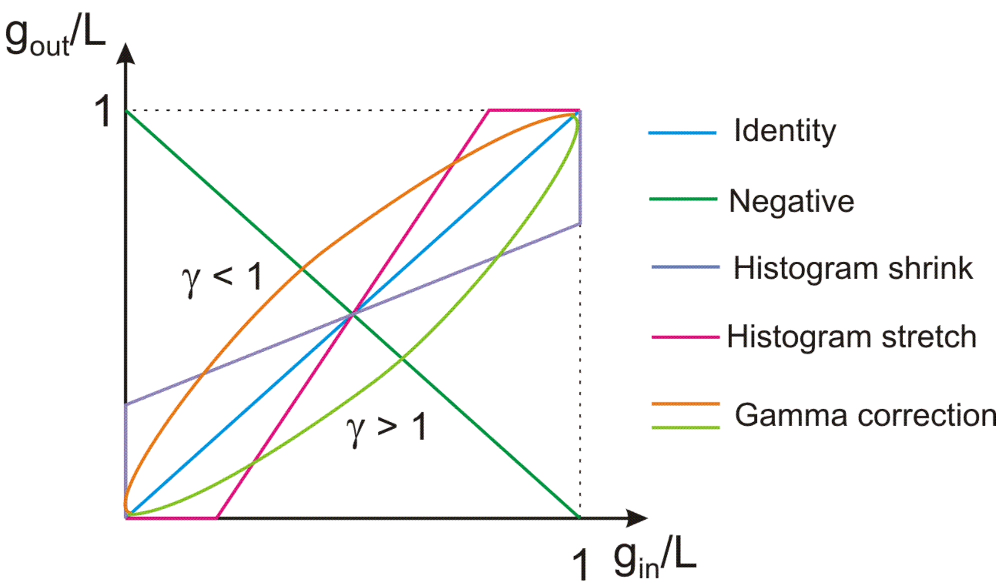
← stretch

Grayscale levels remapping using a transformation function

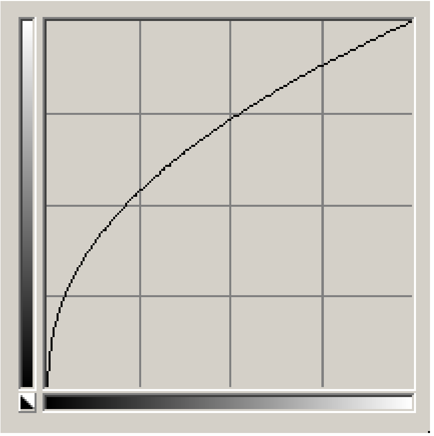
$g_{output} = T\ (g_{input})$

Ex. - gamma correction:

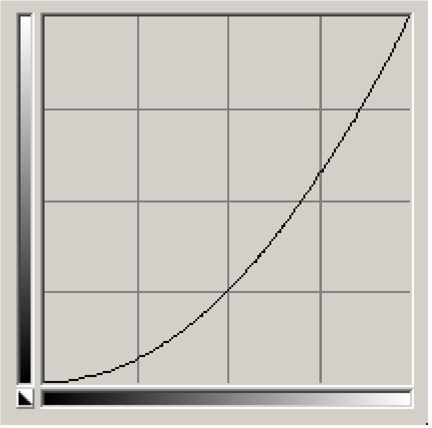
$$g_{out} = c \cdot g_{in}^{\gamma} \qquad \text{sau} \qquad \frac{g_{out}}{L} = c \cdot \left(\frac{g_{in}}{L}\right)^{\gamma}$$



Ex. - gamma correction:



γ< 1: codificare/compresie gamma



γ> 1: decodificare/decompresie gamma

Trasaturi statistice / *Statistical features*

Information – information associated to the gray-level g:

$$I_g = -\log_2 p(g) \qquad [bits]$$

$\Rightarrow$ information is high for a gray level with smaller probability

Entropy – average information from the image:

$$H = -\sum_{g=0}^{L} p(g) \cdot \log_2 p(g) \qquad [bits]$$

$\Rightarrow$ No. bits necessary to encode the gray-levels from the image

H (big) – grayscale values are widely spread

$H_{max} = \log_2 L$ [bits] (uniform PDF)

Energy – how are the gray-levels distributed:

$$E = \sum_{g=0}^{L} [p(g)]^2$$

E (mica) – large number of gray-levels

$E_{max} = 1$ (1 gray-level)

Histogram equalization/ Egalizarea histogramei

Normalized gray-levels:

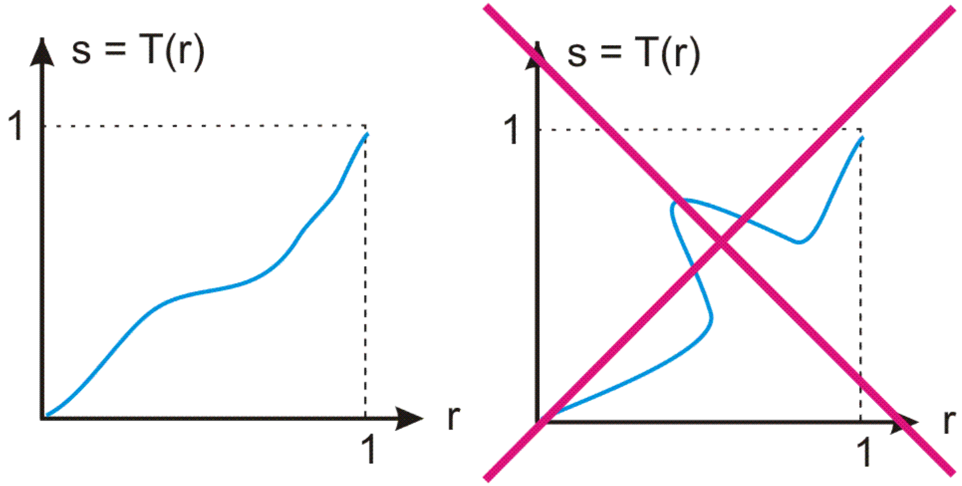
$g \in [0 \ldots L] \Rightarrow r \in [0 \ldots 1]$

Transformation functions:

$s = T(r)$

(a). Bijective and monotonically increasing $\Rightarrow \exists\ r = T^{-1}(s)$

(b). $0 <= T(r) <= 1$



Procesari pe histograma / *Histogram processing*

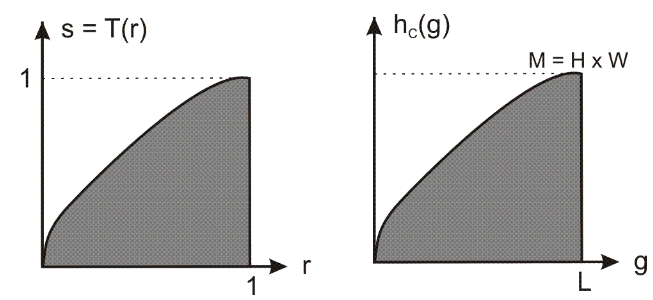
Histogram equalization/ Egalizarea histogramei

- $p_r(r)$, $p_s(s)$ – FDP of the source and destination image
- $p_r(r)$, $T(r)$ are known and T^{-1} satisfies condition (a)

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \qquad (1)$$

Historama cumulativa / FDP cumulativa (CDF)

$$s = T(r) = \int_0^r p_r(w)dw \qquad (2)$$



T satisfies (a) & (b)

Leibniz Rule: the derivative of an integral function superiorly defined is the integral function evaluated in the upper limit:

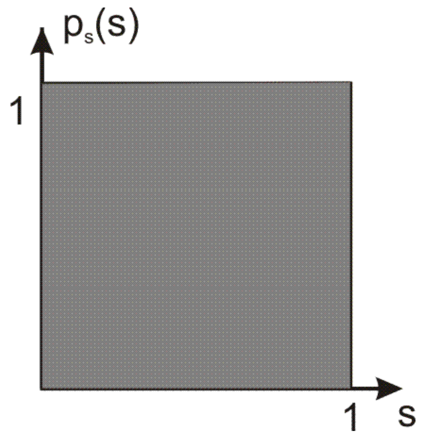
$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = p_r(r) \qquad (3)$$

Technical University of Cluj Napoca
Computer Science Department

Histogram equalization/ Egalizarea histogramei

(1) + (3) $\Rightarrow$

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\left|\frac{1}{p_r(r)}\right| = 1 \quad , \quad 0 <= s <= 1$$



$p_s(s)$:

- uniform PDF
- independent by $p_r(r)$

Histogram equalization algorithm

$$p_r(r_k) = \frac{n_k}{n} \quad , \quad k = 0...L, \quad r_k = k / L$$

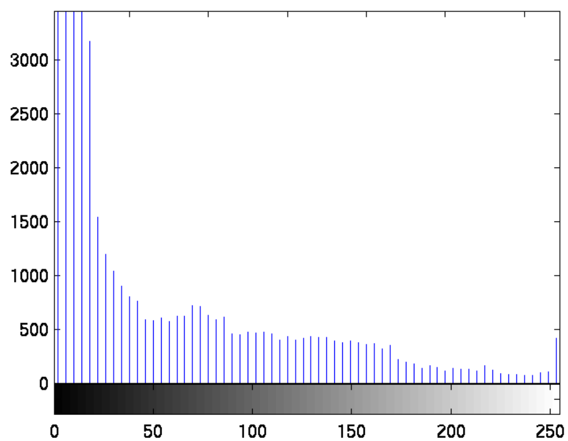
$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n} \quad , \quad k = 0...L$$

$\Rightarrow$ remapping the input image gray-levels: r_k -> s_k

Procesari pe histograma / *Histogram processing*

Histogram equalization/ Egalizarea histogramei: rezultate

```
I = imread('tire.tif');
J = histeq(I);
imshow(I)
figure, imshow(J)
figure; imhist(I,256)
figure; imhist(J,256)
```



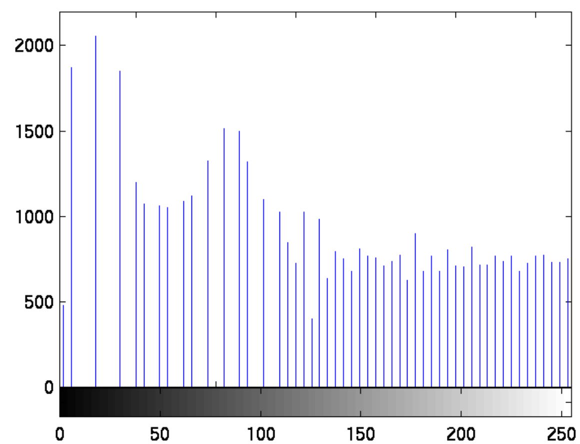


Image filtering (space domain filters)

Filtrarea imaginilor (filtre spatiale)

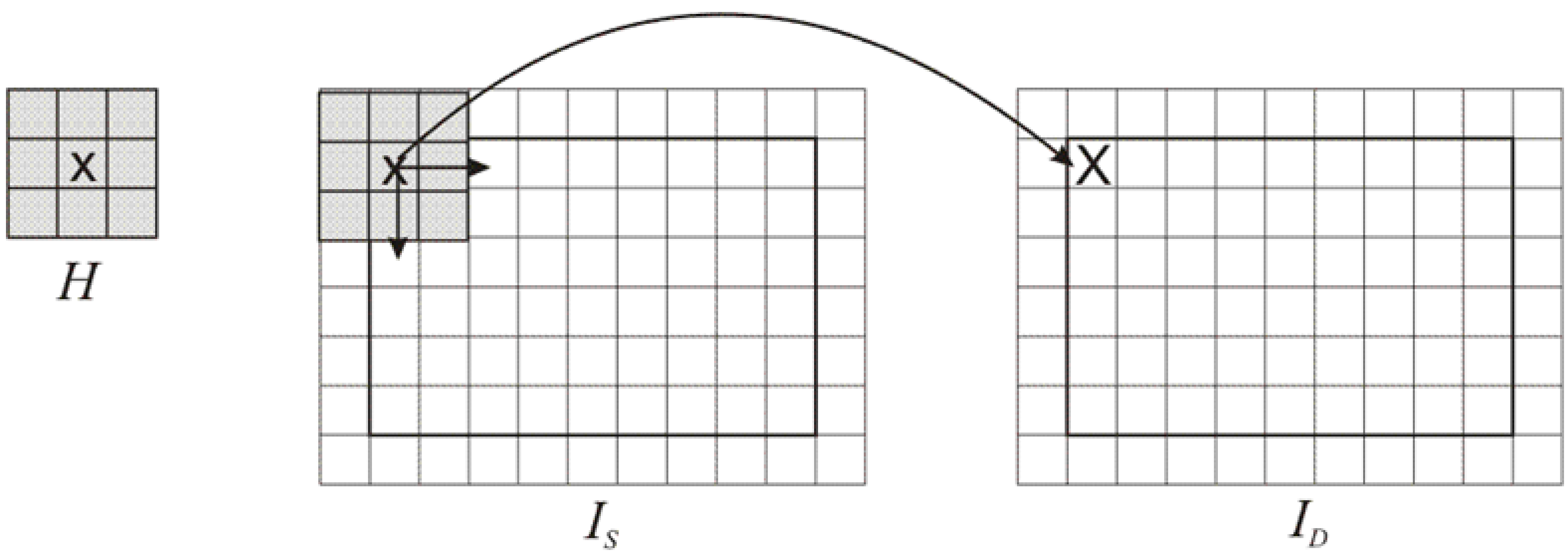
Convolution

This operator is the basis of the linear image filtering operation applied in the spatial image domain (by directly manipulating image pixels)

It implies the usage of a convolution kernel H (usually squared shape, by size w*w, with width/height = w=2k+1) which is applied on the image using a *shift & multiply* scheme:

$$I_D = H * I_S$$

$$I_D(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} H(i,j) \cdot I_S(x+i, y+j), \quad x = 0..Height-1, \quad y = 0..Width-1$$



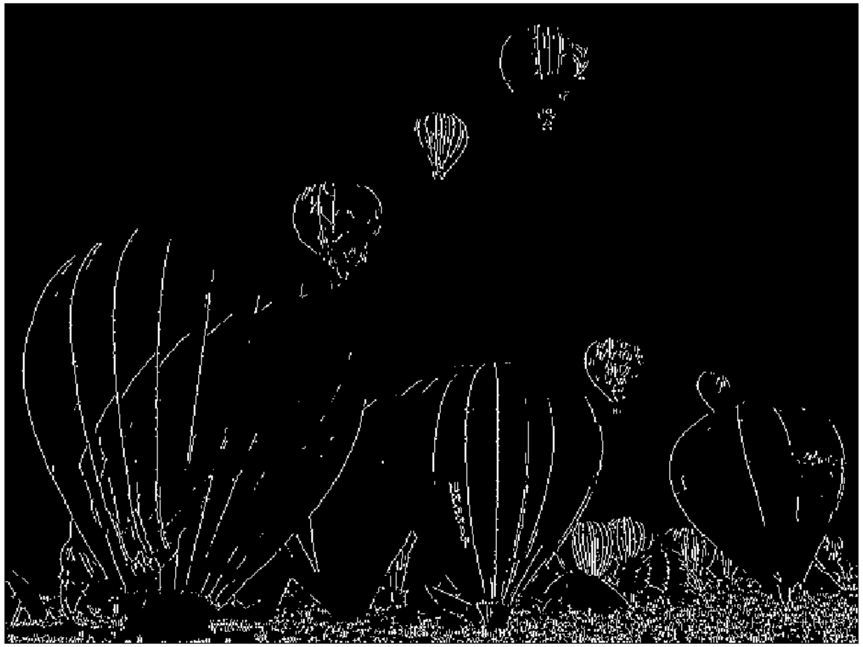
Convolutie - exemple

Example: filter for vertical edges detection (computes the horizontal derivative of the image)

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

X

Y = filter2(H,X)



Technical University of Cluj Napoca
Computer Science Department

Filtre de tip trece-jos / low-pass filters

Used for smoothing the gray-levels / noise reduction. The outcome is an averaging operation of the current pixel by his neighbor's values ⇒ "blur" effect on the image

These filters have only positive elements. From that reason the result of the convolution is normalized by dividing it with the sum of the filters' coefficients:

$$I_D(x,y) = \frac{1}{c} \cdot \sum_{i=-k}^{k} \sum_{j=-k}^{k} H(i,j) \cdot I_S(x+i, y+j)$$

$$c = \sum_{i=-k}^{k} \sum_{j=-k}^{k} H(i,j)$$

Average (mean) filter (3x3):

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian filter (3x3):

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

Filtre de tip trece-jos / low-pass

a. Original image

b. Result obtained by applying a 3x3 mean filter.

c. Result obtained by applying a 5x5 mean filter.

Technical University of Cluj Napoca
Computer Science Department

Filtre de tip trece-sus / high-pass

Image regions / pixel with local intensity variations are highlighted (i.e. edge pixels). The outcome is a high pass filtering (image sharpening)

The filters / kernels can have positive an negative elements. Edge detection kernels must have a null sum:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

a. The result of applying the Laplace edge detection filter on the original image

b. The result of applying the Laplace edge detection filter on the blurred image

c. The result obtained by filtering the original image with the high-pass filter

Technical University of Cluj Napoca
Computer Science Department

Filtre de tip trece-sus / high-pass

High-pass filters will have both positive and negative coefficients. You must ensure that the final result is an integer between 0 and 255! There are three possibilities to ensure that the resulting image fits the destination range.

1. The first one is to compute:

$$S_+ = \sum_{F_k > 0} F_k, \quad \mathrm{S}_- = \sum_{F_k < 0} -F_k,$$

$$S = \frac{1}{2\max\{S_+, S_-\}}$$

$$I_D(u,v) = S(F * I_S)(u,v) + \left\lfloor \frac{L}{2} \right\rfloor$$

In the formula above represents the sum of positive filter coefficients and the sum of negative filter coefficients magnitudes. This result of applying the high-pass filter always lies in the interval where L is the maximum image gray level (255). The result of this transform will place scale the result to [-L/2, L/2] and then move the 0 level to L/2.

Technical University of Cluj Napoca
Computer Science Department

Filtre de tip trece-sus / high-pass

2. Another approach is to perform all operations using signed integers determine the minimum and maximum and then linearly transform the resulting values according to:

$$D = \frac{L(S - \min)}{\max - \min}$$

3. The third approach is to compute the magnitude of the result and saturate everything that exceeds the maximum level L.

Technical University of Cluj Napoca
Computer Science Department

Noise modeling & removal

Modelarea si eliminarea zgomotelor

Definitia zgomotului / *Noise definition*

Noise := Any process (n) that affects image (f) and is not part of the scene (s):

$$f(i,j) = s(i,j) + n(i,j) \quad \text{(additive noise model)}$$

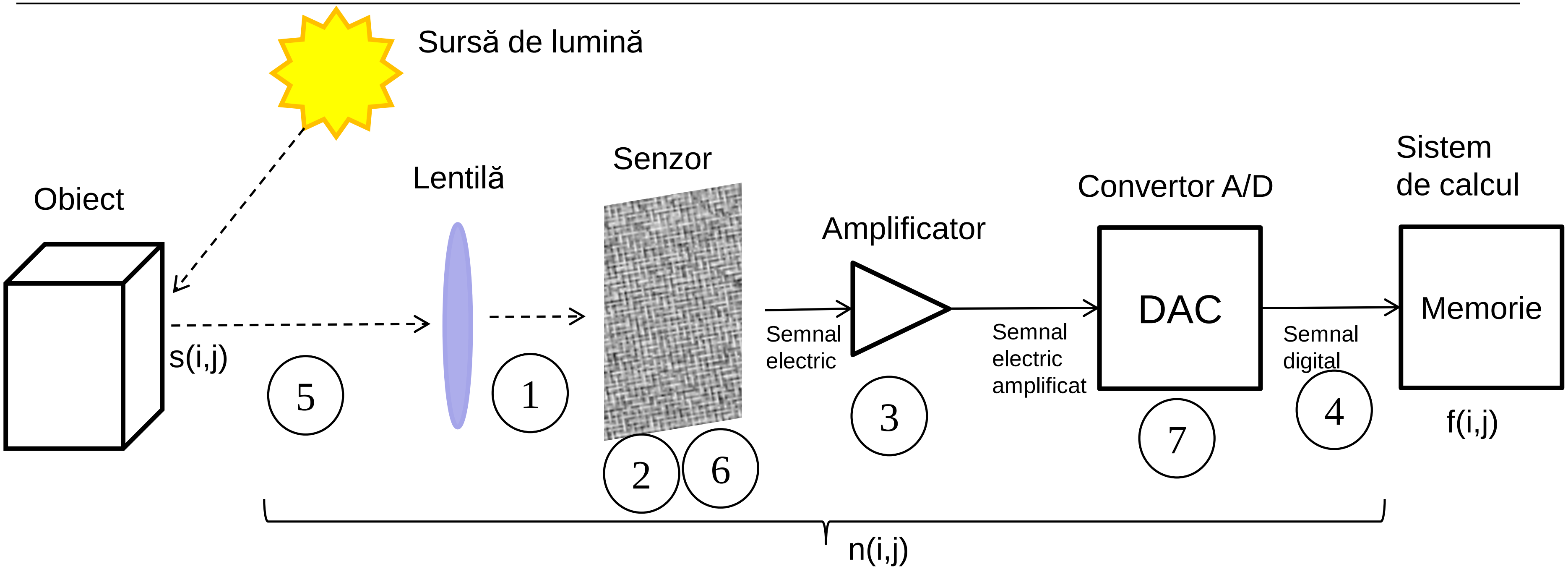
Causes:

1. Discrete nature of the radiation
2. Detector sensitivity (variable sensitivity of the sensorial elements CCD/CMOS $\Rightarrow$ fixed pattern noise (dark current noise (DCN) & photon response non-uniformity (PhRNU))
3. *Electrical noise*
4. *Data transmission errors*
5. *Air turbulences*
6. *Spatial resolution of the sensor*
7. *Quantization resolution of the color /gray levels*

Types of noise (FDP p(g) shape):

- *Salt&pepper (sare si piper)*
- *Uniform*
- *Gaussian*
- *Other distributions: Rayleigh, Erlang/Gamma, Exponential etc.*

Surse de zgomot



s(i,j) – semnalul iniţial, lumina reflectată de pe obiect
f(i,j) – semnalul (imaginea digitală) memorat în sistemul de calcul
n(i,j) – zgomot, procese care se interpun între s şi f (1…7)

1 – Natura discretă a radiaţiei
2 – Sensibilitatea variabilă a elementelor (pixelilor) senzorului
3 – Zgomotul electric
4 – Erori de transmisie a datelor

5 – Turbulenţe atmosferice
6 – Rezoluţia senzorului (erori de cuantizare spaţială)
7 – Rezoluţia convertorului A/D (erori de cuantizare a semnalului analogic)

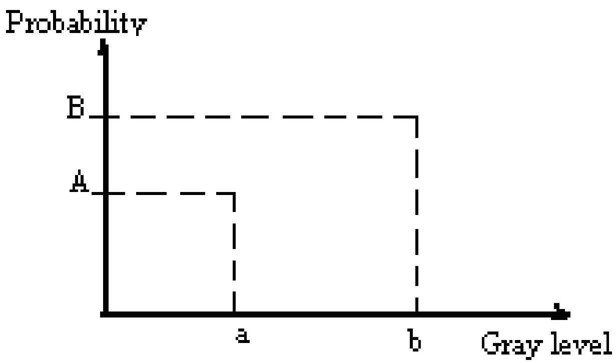
Salt &Pepper noise (sare si piper)

Causes

- Malfunctioning of sensor's cells
- Malfunctioning of memory cells
- Synchronization errors in the signal digitization process
- Bits loss on the communication channel

Model

$$PDF_{salt\,\&\,pepper} = \begin{cases} A & for\ g = a\ ("pepper") \\ B & for\ g = b\ ("salt") \end{cases}$$



Salt & pepper noise

In the salt&pepper noise model only two possible values are possible, a and b, and the probability of obtaining each of them is less than 0.1 (otherwise, the noise would vastly dominate the image). For an 8 bit/pixel image, the typical intensity value for pepper noise is close to 0 and for salt noise is close to 255.

Salt &Pepper noise (sare si piper)

Technical University of Cluj Napoca
Computer Science Department

Salt &Pepper noise (sare si piper)

Eliminating the Salt&Pepper noise ⇒ Median filter (non-linear filter)

Ordered filters are based on a specific image statistic, called ordered statistic. They are called non-linear, because they cannot be applied as a linear operator (such as a convolution kernel). These filters operate on small windows, and replace the value of the central pixel (similarly to convolution). The ordered statistic is a technique which arranges all the pixels in sequential order, based on their gray-level value. The position of an element in this ordered set can be characterized by its rank. Given a NxN window W, the pixel values can be sorted in ascending order:

$$I_1 \leq I_2 \leq I_3 \leq \ldots \leq I_{N^2}$$

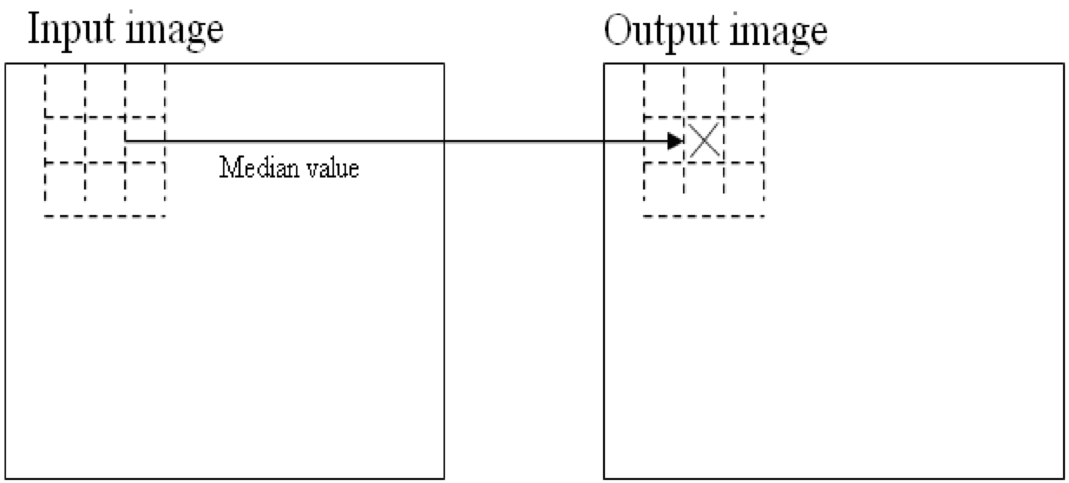
$\{ I_1, I_2, I_3, \ldots, I_{N^2} \}$ represent the intensity values of the pixels located within the *N*x*N* window *W*

$$\begin{bmatrix} 110 & 110 & 114 \\ 100 & 104 & 104 \\ 95 & 88 & 85 \end{bmatrix} \Rightarrow \{85, 88, 95, 100, 104, 104, 110, 110, 114\}$$

SVR

Salt &Pepper noise (sare si piper)

The median filter: selects the middle value from the ordered statistic and replaces the destination pixel with it. In the example above, the selected value would be 104. The median filter allows the elimination of *salt&pepper noise*.

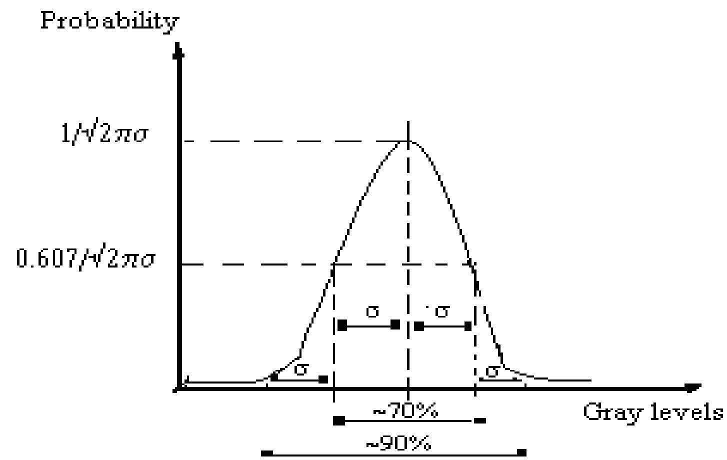


Gaussian noise

- Gaussian noise is useful for modeling natural processes which introduce noise (e.g. noise caused by the discrete nature of radiation and the conversion of the optical signal into an electrical one – detector/shot noise, the electrical noise during acquisition – sensor electrical signal amplification, etc.).
- For modelling these types of noises a poisson distribution should be used but it is to complicated to handle it mathematically $\Rightarrow$ can be approximated by a Gaussian distribution

Model

$$FDP_{Gaussian} = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$



Gaussian noise

where:

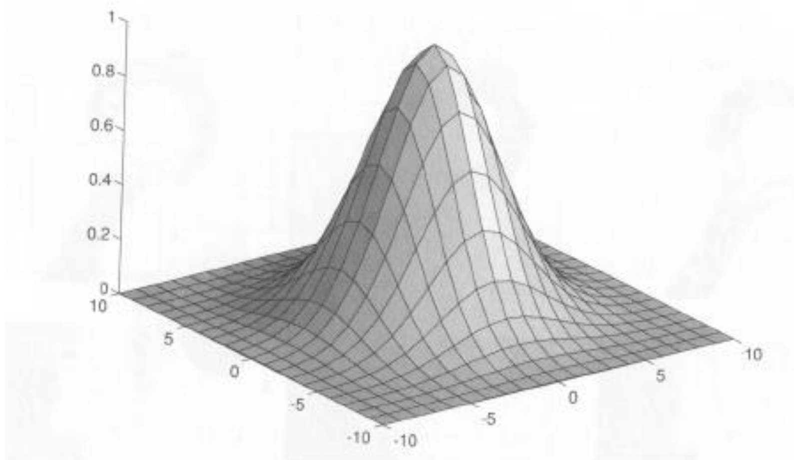
g = gray level;

μ = media zgomotului;

σ = deviatia standard a zgomotului;

Gaussian noise

Design a variable size Gaussian kernel

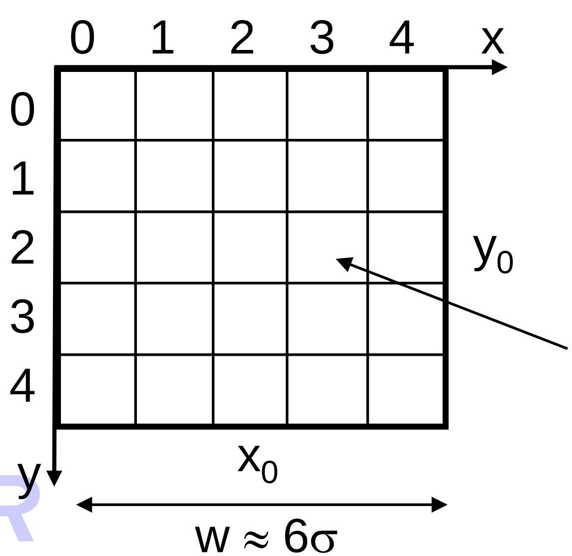


FDP for gaussian noise with 0 mean:

$$G(x,y) = G(x)G(y)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

The filter size *w* of such a filter is usually 6σ (for example, for a Gaussian noise with **σ=0.8 $\Rightarrow$ w = 4.8 $\approx$ 5).**



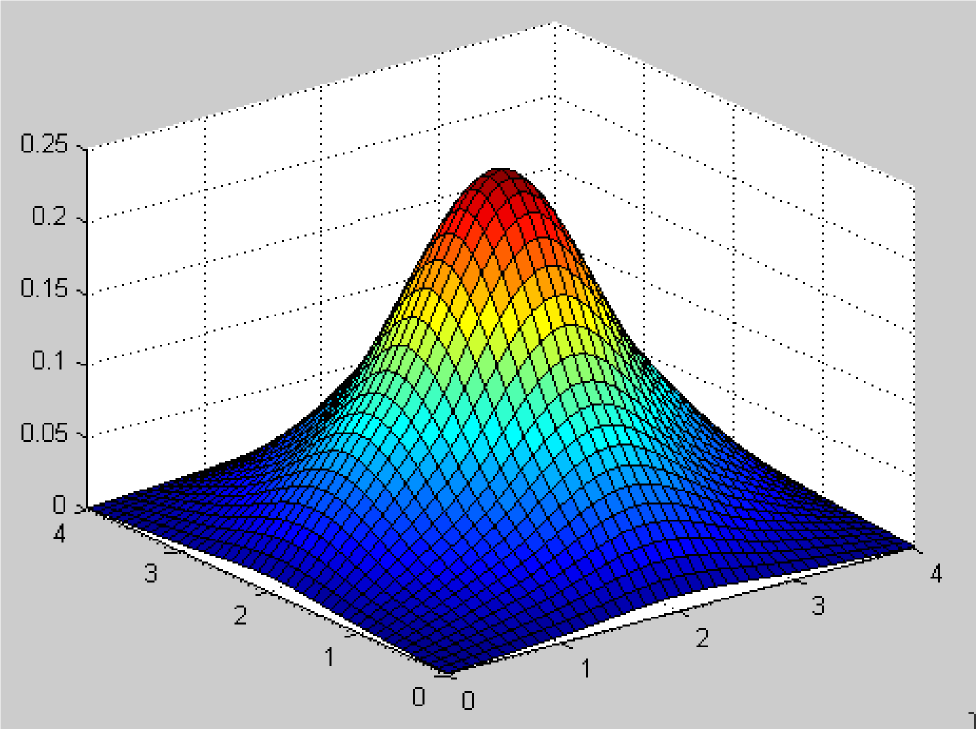
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-((x-x_0)^2+(y-y_0)^2)}{2\sigma^2}}$$

Matlab example:

```
function [G]=gaussian(sigma);
w=round(6*sigma);
x0=floor(w/2)+1;
y0=x0;
sigma2=2*sigma*sigma;
for x=1:w
 for y=1:w
  G(x,y)=1/(pi*sigma2)*exp(-((x-x0)*(x-x0)+(y-y0)*(y-y0))/sigma2);
 end
end
[X,Y] = meshgrid(1:.1:w,1:.1:w);
Z = interp2(G,X,Y,'cubic');
x=1:0.1:5
y=1:0.1:5;
surf(x,y,Z);
```

$$\sigma = 0.8 \Rightarrow w = 5$$

```
G =
    0.0005    0.0050    0.0109    0.0050    0.0005
    0.0050    0.0521    0.1139    0.0521    0.0050
    0.0109    0.1139    0.2487    0.1139    0.0109
    0.0050    0.0521    0.1139    0.0521    0.0050
    0.0005    0.0050    0.0109    0.0050    0.0005
>> sum(sum(G))
ans =
    0.9982
```



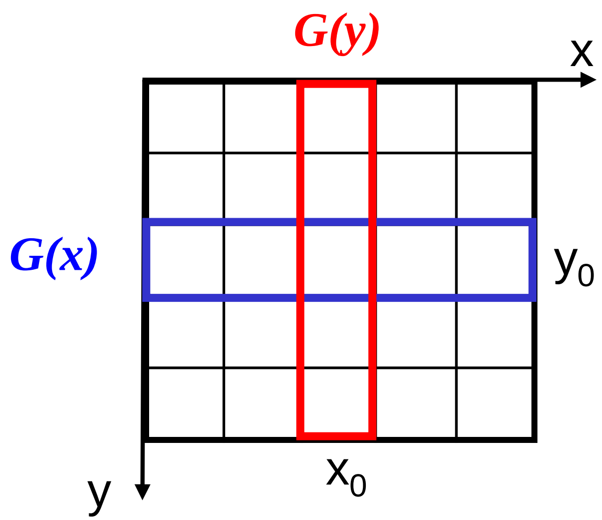
Filtering the Gaussian noise

Spatial domain filters (using convolution kernels)

$$I_D(x,y) = G(x,y) * I_S(x,y)$$

sau

$$I_D(x,y) = \big(G(x)G(y)\big) * I_S(x,y) = G(x) * \big(G(y) * I_S(x,y)\big)$$



$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-x_0)^2}{2\sigma^2}}$$

$$G(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y-y_0)^2}{2\sigma^2}}$$

Technical University of Cluj Napoca
Computer Science Department

Signal to Noise Ratio – SNR (Raportul semnal zgomot)

Additive noise model:

$$f(i,j) = s(i,j) + n(i,j)$$

n – zero mean $(\langle n(i,j)\rangle = 0)$ and signal independent $(\langle s(i,j)n(i,j)\rangle = 0)$ $\Rightarrow$

$$\langle s(i,j)\rangle = \langle f(i,j)\rangle = \mu$$

$$\sigma_f^2 = \sigma_s^2 + \sigma_n^2 \qquad (1)$$

$\Rightarrow$ Noise alters only the standard deviation and not the mean of the image:

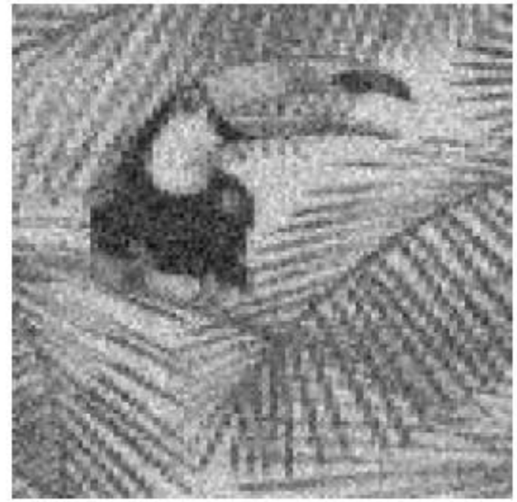
$$SNR = \frac{\sigma_s}{\sigma_n} = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$$

SNR	$>$	20	Little visible noise
SNR	$\approx$	10	Some noise visibile
SNR	$\approx$	4	Noise clearly visible
SNR	$\approx$	2	Image severly degraded
SNR	$\approx$	1	Is there an image?

Technical University of Cluj Napoca
Computer Science Department

SNR – Exemples:

SNR = 1



SNR = 2

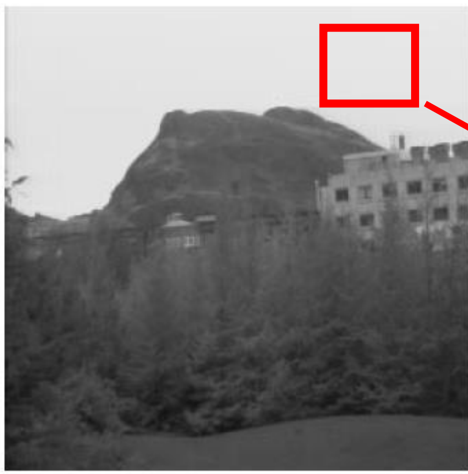
SNR = 4

SNR = 8

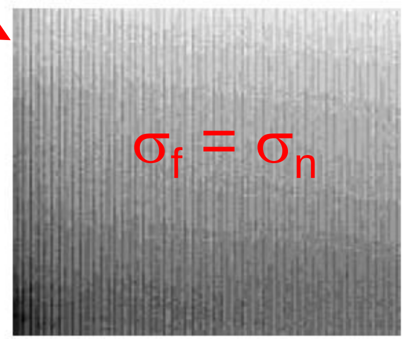
Computing SNR

Single image case

1. Compute σ_f on the whole image
2. Select a ROI with uniform intensity σ_S =0 (ex: sky, water, wall etc.) and compute $\sigma_f = \sigma_n$



Whole Image



Piece of Sky

$$SNR = \frac{\sigma_s}{\sigma_n} = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$$

Computing SNR

2 images case (successive images of a static scene):

$$f(i,j) = s(i,j) + n(i,j)$$
$$g(i,j) = s(i,j) + m(i,j)$$

- n and m have the same FDP: same mean (0) and standard deviation
- n and m are uncorrelated (independent) with the signal: $(\langle s(i,j)n(i,j)\rangle = 0, \langle s(i,j)m(i,j)\rangle = 0)$
- f and g are uncorrelated $(\langle f(i,j)g(i,j)\rangle = 0)$

$$r = \frac{\langle (fg - \langle f\rangle\langle g\rangle)\rangle}{[\langle |f - \langle f\rangle|^2\rangle\langle |g - \langle g\rangle|^2\rangle]^{1/2}}$$

Normalized correlation between f and g

$$r = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2}$$

$$SNR = \sqrt{\frac{r}{1-r}}$$

Technical University of Cluj Napoca
Computer Science Department

Features detection: edges & corners

Detectia de trasaturi: Detectia punctelor de muchie. Detectia de colturi.

Motivation

Purpose of edge detection?

- It seems that human visual system uses edges as primitives in the perception/recognition process (complex information (color and texture) are inferred afterwards)

- It is possible to recognize shapes/objects only based on contours (i.e. caricatures, bw comics / cartoons).

$\Rightarrow$ Edge detection is an important step in the automated image analysis process

$\Rightarrow$ Edge detection $\approx$ segmentation process

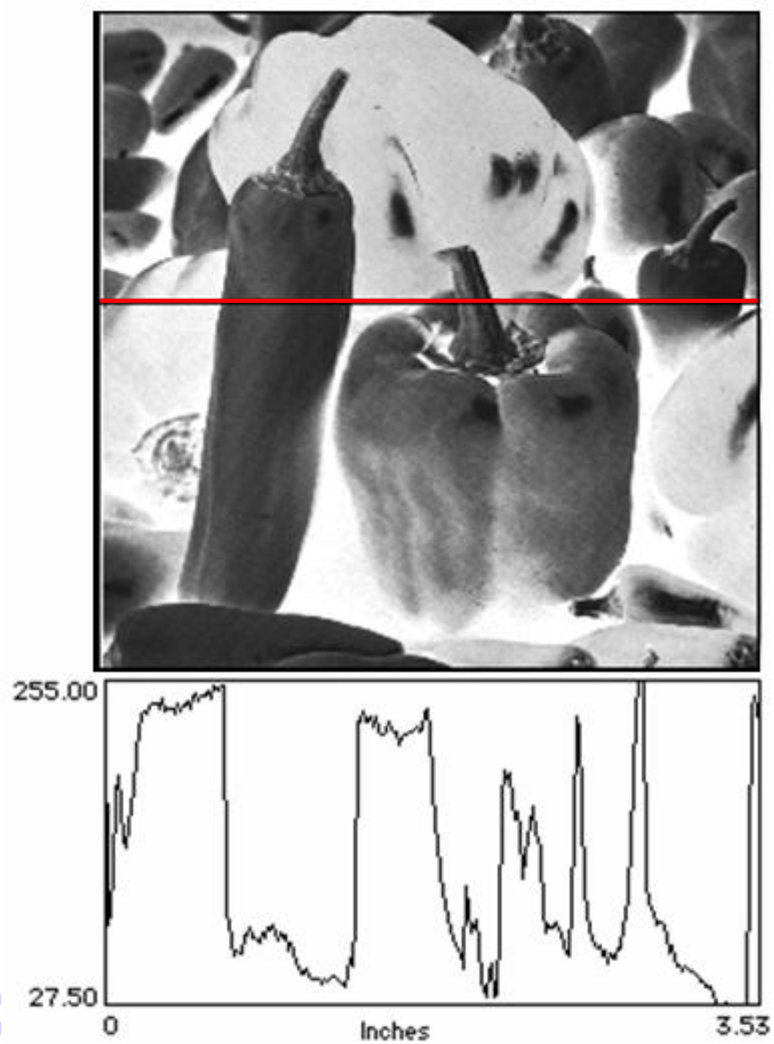
Segmentation := from low level information (pixels / row data) $\Rightarrow$ high level information is extracted:

- edge points $\Rightarrow$ contours $\Rightarrow$ shape features $\Rightarrow$ analysis

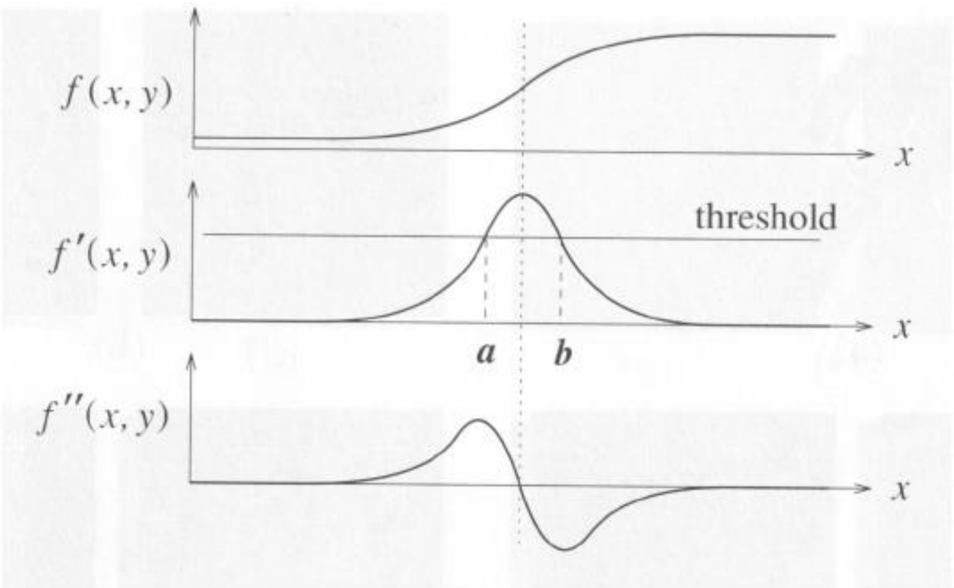
- edge points $\Rightarrow$ features for sparse stereo reconstruction

Definition

Muchie / *edge* := the frontier that separates 2 regions of different brightness (usual the brightness has an abrupt variation at the edge)



How can we detect an edge ?



Edge points intensity profile

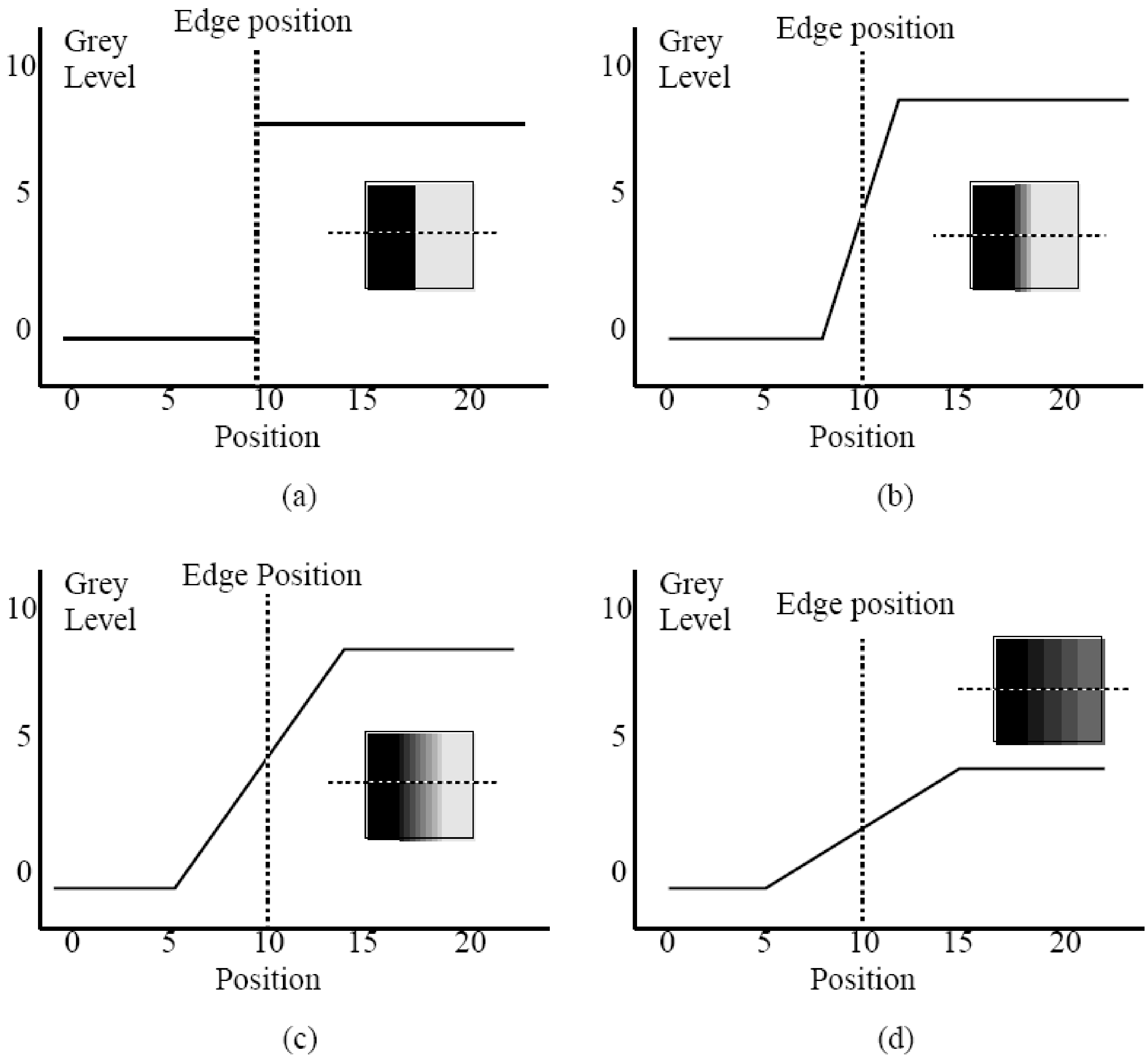
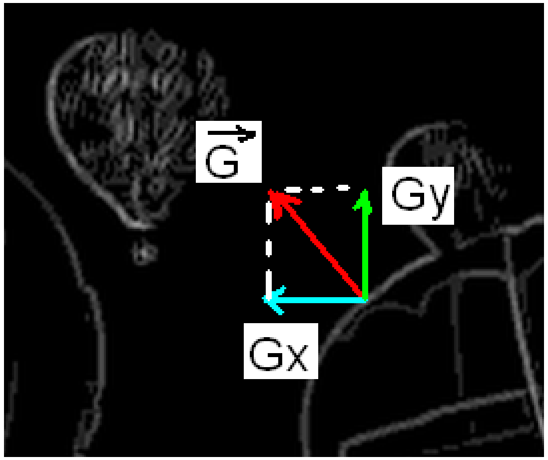


Image gradient (1-st order derivative)

Gradient of a 2D function (*image gradient*)

$$G[f(x,y)] = \begin{bmatrix} G_{fx} \\ G_{fy} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\[2ex] \dfrac{\partial f}{\partial y} \end{bmatrix}$$

$$= \begin{bmatrix} \lim\limits_{\Delta x \to 0} \dfrac{f(x+\Delta x, y) - f(x,y)}{\Delta x} \\[2ex] \lim\limits_{\Delta y \to 0} \dfrac{f(x, y+\Delta y) - f(x,y)}{\Delta y} \end{bmatrix}$$



For a digital image: $\Delta x = \Delta y = 1$

$$G[f[i,j]] = \begin{bmatrix} f[i+1,j] - f[i,j] \\ f[i,j+1] - f[i,j] \end{bmatrix} \Rightarrow \quad G_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Approximations of gradient

Operatorul Roberts

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

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

Operatorul Prewitt

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

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

Operatorul Sobel

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

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

SVR

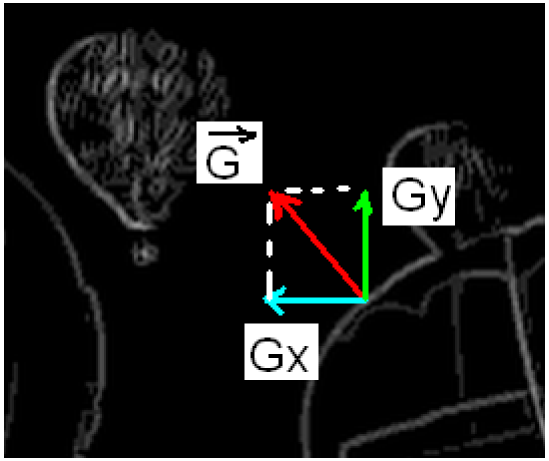
Gradient features

Magnitude

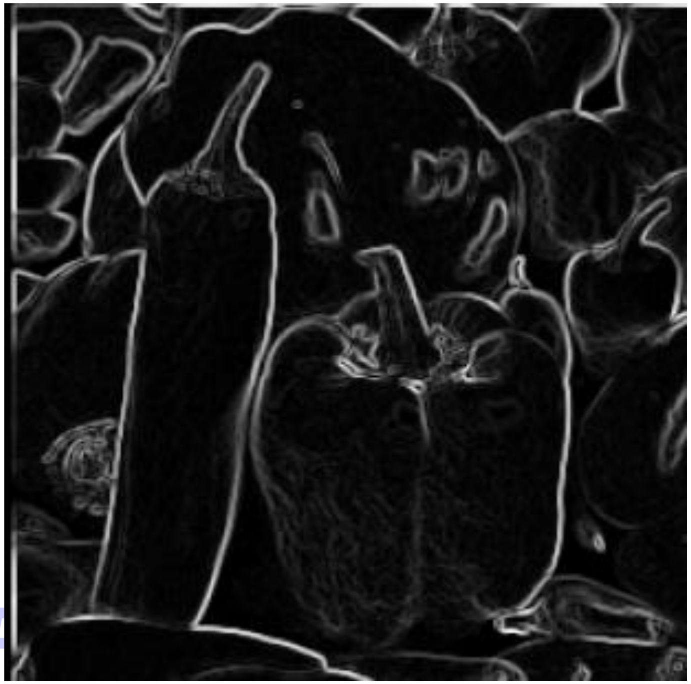
$$|G| = \sqrt{G_{fx}^2 + G_{fy}^2}$$

Directiion

$$dir = arctg\left(\frac{G_{fy}}{G_{fx}}\right)$$



Imaginea |G|

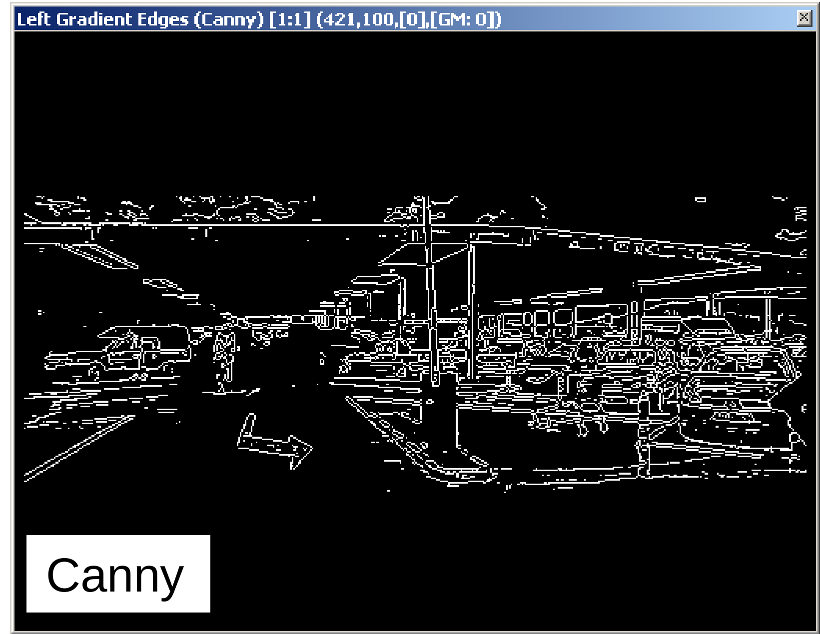


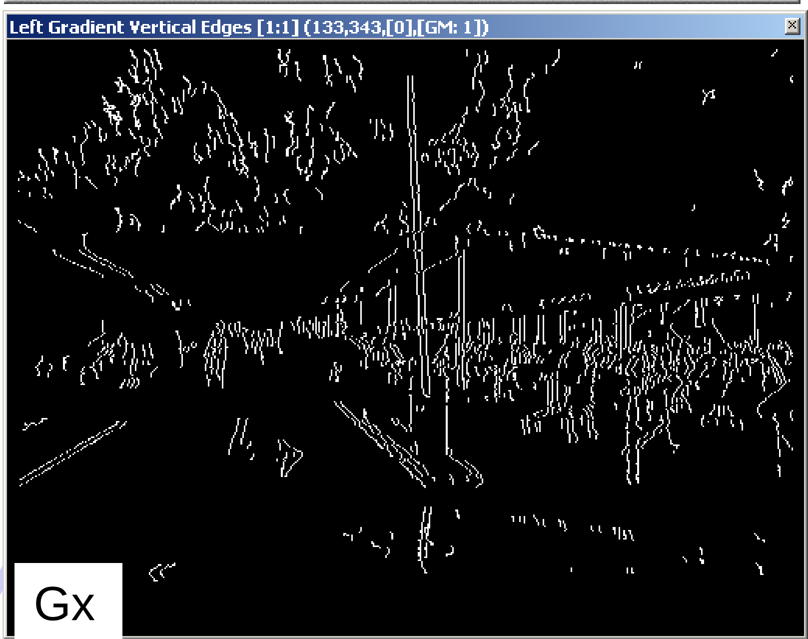
$\Rightarrow$

Thresholding with **T**

Examples


Left Image [1:1] (379,0,[253],[GM: 0])


Left Gradient Edges (Canny) [1:1] (421,100,[0],[GM: 0])
Canny


Left Gradient Vertical Edges [1:1] (133,343,[0],[GM: 1])
Gx


Left Horizontal Edges [1:1] (284,0,[0],[GM: 0])
Gy

Canny edge detection method

Features of the Canny edge detector

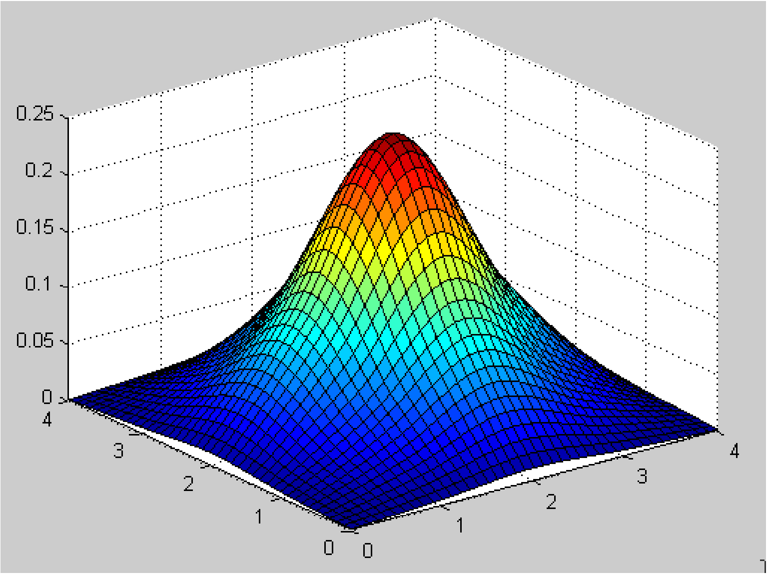
- Maximizes the signal to noise ratio for a correct
- Good localization of the edge
- Minimization of the positive responses to a single edge (non-edges elimination)

Algorithm

1. Gaussian filtering
2. Edge magnitude & direction computation
3. Non-maxima suppression (edge thinning)
4. Hysteresis thresholding (edge linking)

Technical University of Cluj Napoca
Computer Science Department

1. Gaussian filtering



$$g(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{((x-x_0)^2+(y-y_0)^2)}{2\sigma^2}}$$

$$g(x,y)=g(x)\cdot g(y)$$

$$g(x)=\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\cdot e^{-\frac{(x-x_0)^2}{2\cdot\sigma^2}}$$

$$g(y)=\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\cdot e^{-\frac{(y-y_0)^2}{2\cdot\sigma^2}}$$

Example: σ = 0.8 $\Rightarrow$ w = 5 (filter dimension) $w\approx 6\cdot\sigma$

G(x,y) =

[0.0005	0.0050	0.0109	0.0050	0.0005
0.0050	0.0521	0.1139	0.0521	0.0050
0.0109	0.1139	0.2487	0.1139	0.0109
0.0050	0.0521	0.1139	0.0521	0.0050
0.0005	0.0050	0.0109	0.0050	0.0005]

$$f(x,y)=f_s(x,y)*g(x,y)=\left(f(x,y)*g(x)\right)*g(y)$$

Technical University of Cluj Napoca
Computer Science Department

2. Edge magnitude & direction

$$G[f(x,y)] = \begin{bmatrix} G_{fx} \\ G_{fy} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$G_{fx}(x,y) = f(x,y) * S_x(x,y)$$
$$G_{fy}(x,y) = f(x,y) * S_y(x,y)$$

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

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

Magnitude

$$|G| = \sqrt{G_{fx}^2 + G_{fy}^2}$$

Directiion

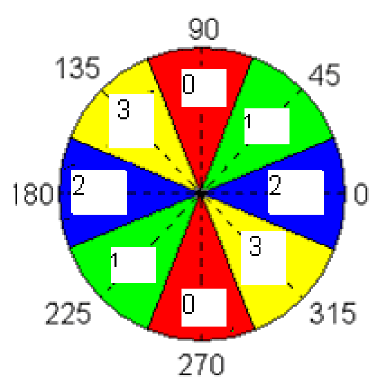
$$dir = arctg\left(\frac{G_{fy}}{G_{fx}}\right)$$

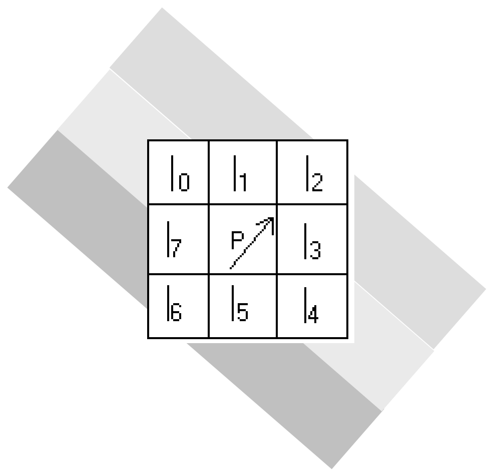
Technical University of Cluj Napoca
Computer Science Department

3. Non-maxima suppression

⇒ Edge thinning along the gradient direction (1 pixel thick)

Quantify the gradient directions:





P is a local maxima if:

$$G_6 < G \text{ and } G_2 < G$$

Where: G, G_2, G_6 are the gradient magnitudes in P, I_2, I_6.

If P is a local maxima is retained.

Otherwise is eliminated.

Technical University of Cluj Napoca
Computer Science Department

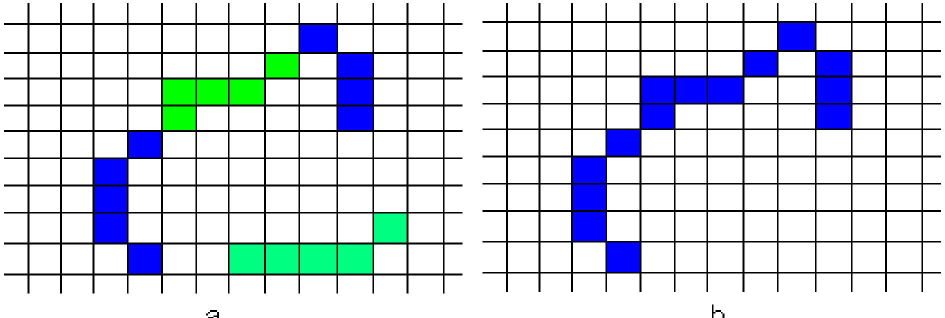
4. Hysteresis thresholding

⇒ Edge linking (contour defragmentation)

1. Two thresholds are used: θ_L (low) and θ_H and the following thresholding scheme is applied:

- Every edge point with magnitude bellow θ_L is labeled as *non-edge*
- Every edge point with magnitude above θ_H is labeled as *strong edge*
- Every edge point with magnitude between θ_L and θ_H is labeled as *weak edge*

2. Apply an algorithm similar with the labelling one that marks weak edge points as strong if they are connected to strong edge points and eliminates weak edge points if they are not connected to strong edge points.



a. Result after step 1: strong (blue) edges and weak (green) edges. b. Result after step 2

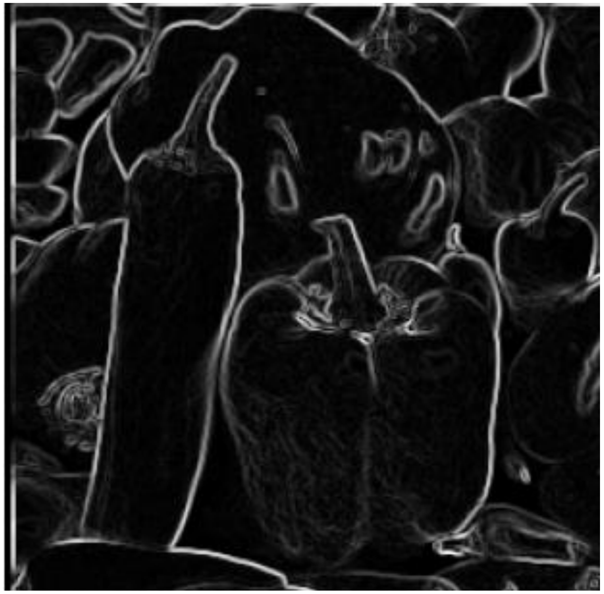
4. Hysteresis thresholding

An efficient implementation of this step uses a queue to perform a breadth first search through WEAK_EDGE points connected to STRONG_EDGE points and mark them as STRONG_EDGE points. The algorithm would look like this:

1. Scan the image, top left to bottom right, pick the first STRONG_EDGE point encountered and push its coordinates in the queue.
2. While (queue is not empty)
    a. Extracts the first point from the queue
    b. Find all the WEAK_EDGE neighbors of the current point
    c. Label in the image all these neighbors as STRONG_EDGE points
    d. Push the image coordinates of these neighbors into the queue
    e. Continue to the next STRONG_EDGE point
3. Go to step 1 considering the next STRONG_EDGE point.
4. Eliminate the remaining WEAK_EDGE points from the image by turning them to NON_EDGE (0)

Results

Magnitude image |G|



⇒

Thresholding with **T**

⇒

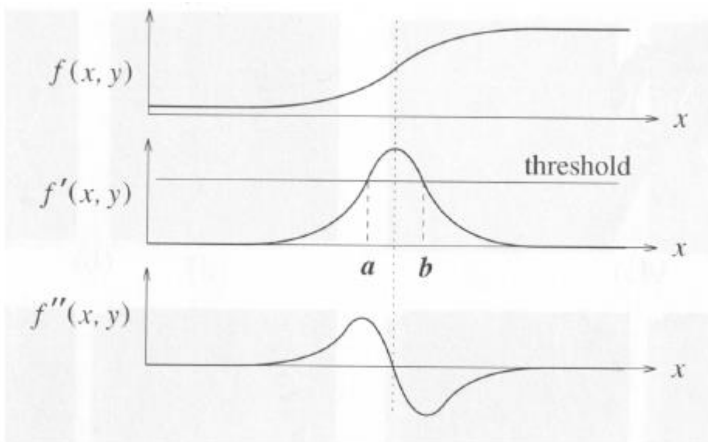
Non-maxima suppression

⇒

Hysteresis thresholding

SVR

2-nd order derivative



$$f''(x) = \frac{f'(x+\Delta x) - f'(x)}{\Delta x}$$

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

Laplacian

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

$$\nabla^2 f(x,y) = f * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplacian of Gaussian (LoG/ Mar-Hilderth)

$$h(x, y) = \nabla^2[g(x, y) \otimes f(x, y)]$$

$$h(x, y) = [\nabla^2 g(x, y)] \otimes f(x, y)$$

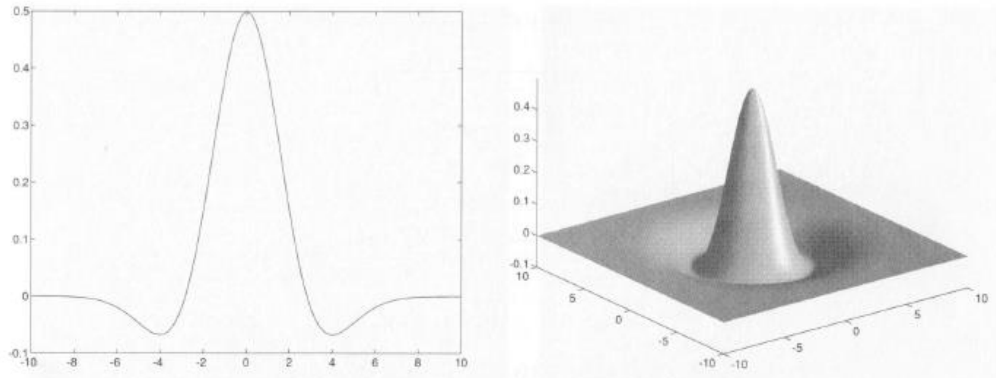


Figure 5.11: The *inverted* Laplacian of Gaussian function, $\sigma = 2$, in one and two dimensions.

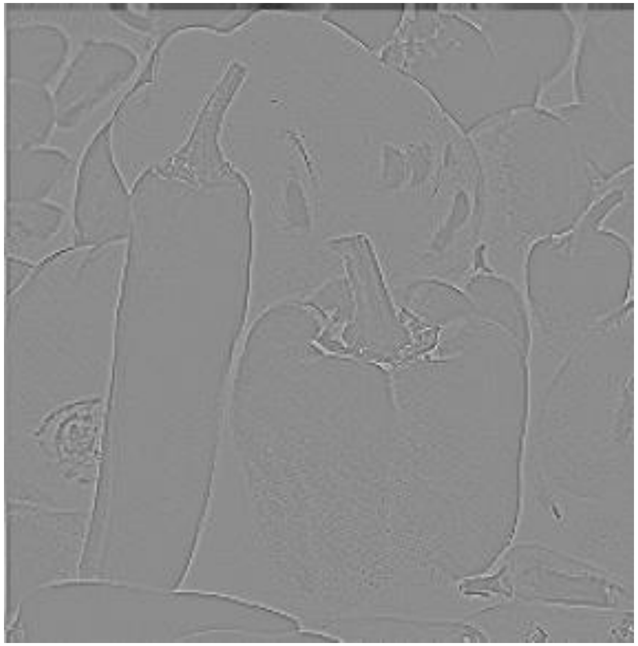
$$LoG(x, y) = \nabla g(x, y) = \frac{\partial^2 g(x)}{\partial x^2} \cdot g(y) + g(x) \cdot \frac{\partial^2 g(y)}{\partial y^2}$$

$$LoG(x, y) = \frac{-1}{2 \cdot \pi \cdot \sigma^6} \cdot (\sigma^2 - x^2) \cdot e^{-\frac{x^2}{2 \cdot \sigma^2}} \cdot e^{-\frac{y^2}{2 \cdot \sigma^2}} +$$

$$+ \frac{-1}{2 \cdot \pi \cdot \sigma^6} \cdot e^{-\frac{x^2}{2 \cdot \sigma^2}} \cdot (\sigma^2 - y^2) \cdot e^{-\frac{y^2}{2 \cdot \sigma^2}}$$

$$LoG(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{2 \cdot \pi \cdot \sigma^6} \cdot e^{-\frac{x^2 + y^2}{2 \cdot \sigma^2}}$$

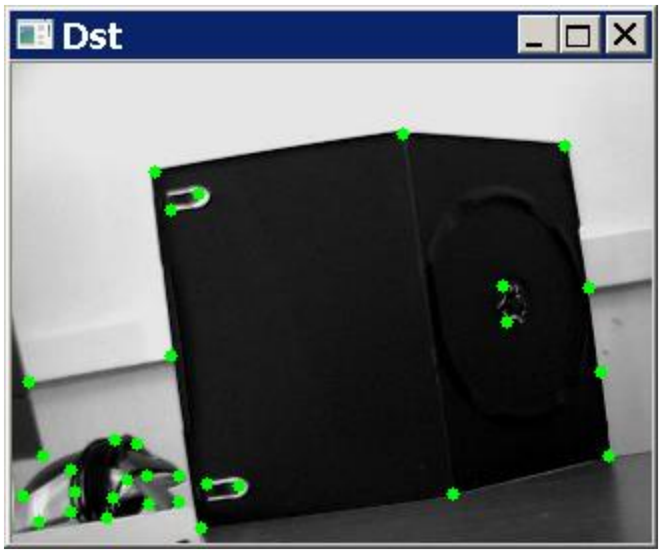
Laplacian of Gaussian (LoG)

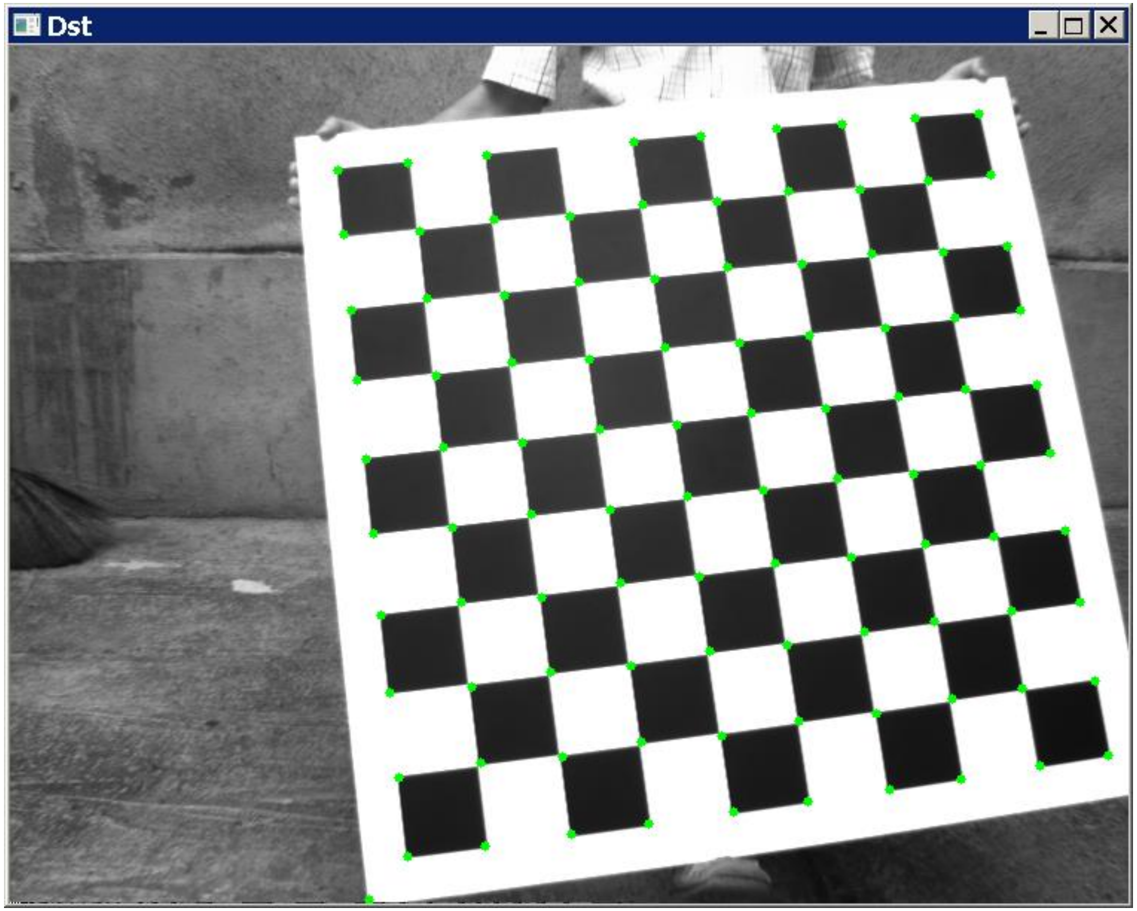


Application: stereo-correlations (compensates for intensity variations between the left and right images / cameras).

Corner detection

Corner := a point where are intensity variations in at least 2 different directions





SVR

Corner detection – *Harris* method

The intensity variation in a point (x,y) for a window *w* shifted with displacement (u,v):

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

Corners $\Rightarrow$ points where E(u,v) has a local maxima

Taylor series aproximation:

$$E(u,v) \approx \sum_{x,y} [I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$E(u,v) \approx \sum_{x,y} u^2 I_x^2 + 2uvI_x I_y + v^2 I_y^2$$

$$E(u,v) \approx [u \quad v] \left(\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

w can be window of Gaussian weights !

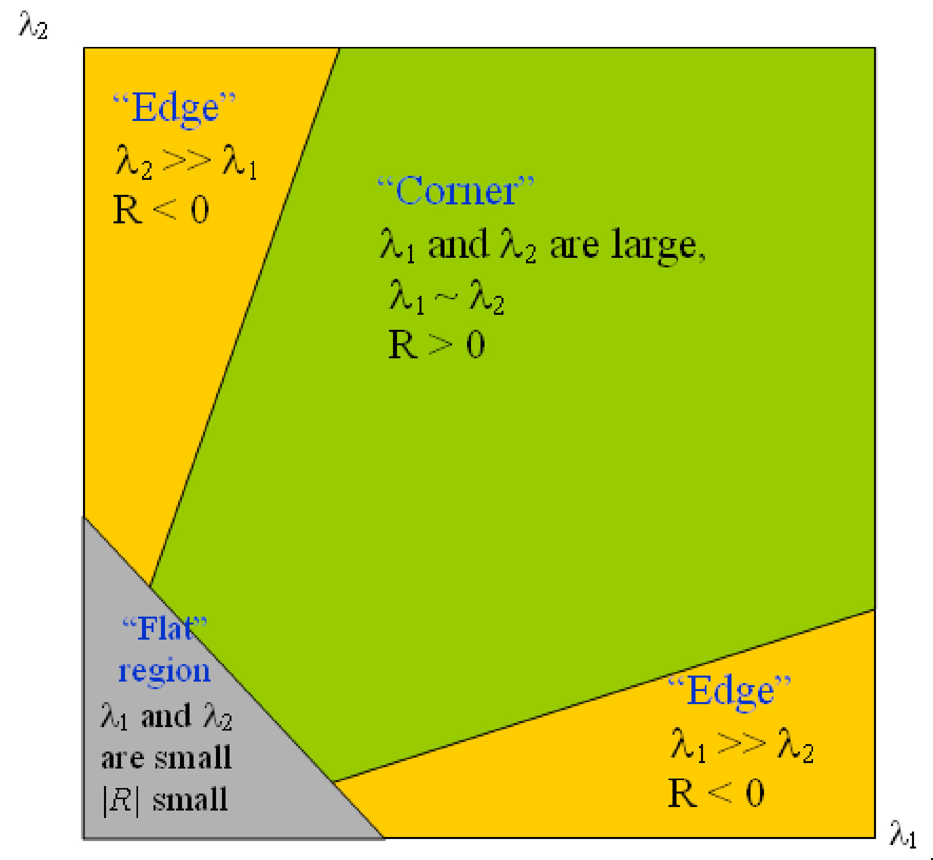
Corner detection – *Harris* method

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

Autocorrelation matrix (covariance of the derivatives) $\Rightarrow$ contains all the differential operators that describe the geometry of the intensity surface in (x,y)

The autocorrelation matrix can be diagonalized by rotating the axes $\Rightarrow$

$$\begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix}, \quad \lambda_{\max} >= \lambda_{\min} >= 0$$



The eigenvalues of M: λ_1 , λ_2 $\Rightarrow$ response function R(x,y) (measure of the "cornerness" in P(x,y))

$$\det(M) = \lambda_1 \lambda_2 = AB - C^2$$

$$trace(M) = \lambda_1 + \lambda_2 = A + B$$

$$R(x,y) = \det(M) - k(trace(M))^2$$

k = 0.04 … 0.15

SVR

Corner detection – *Harris* method

Harris algorithm:

1. For each pixel P(x, y) compute the autocorrelation matrix M.
2. Compute the "map" (matrix) of the response function R(x, y) (in every pixel P(x, y)).
3. Filter out points by thresholding (ex: if. $R < T \Rightarrow R=0$).
4. Apply "non-maximum suppression" $\Rightarrow$ retain only local maxima (others are eliminated: R=0).
5. Al remaining points ($R > 0$) will be the reported corners.
6. Optional you can also limit the maxim no. of reported corners.

[2] A. Koschan, M. Abidi, Digital Color Image Processing, Wiley & Sons, 2008. - cap 6, pag 143 -144