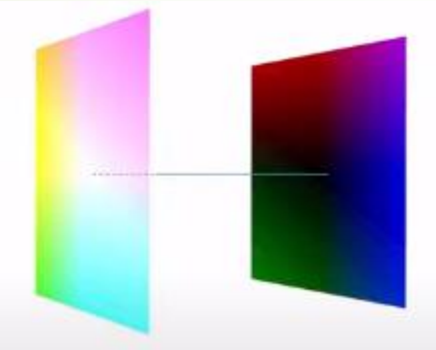
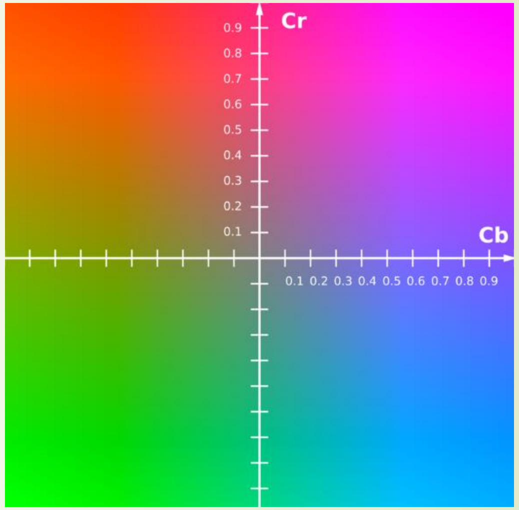
Color Spaces (YCbCr)

YCbCr can be visualized in 3 dimensions



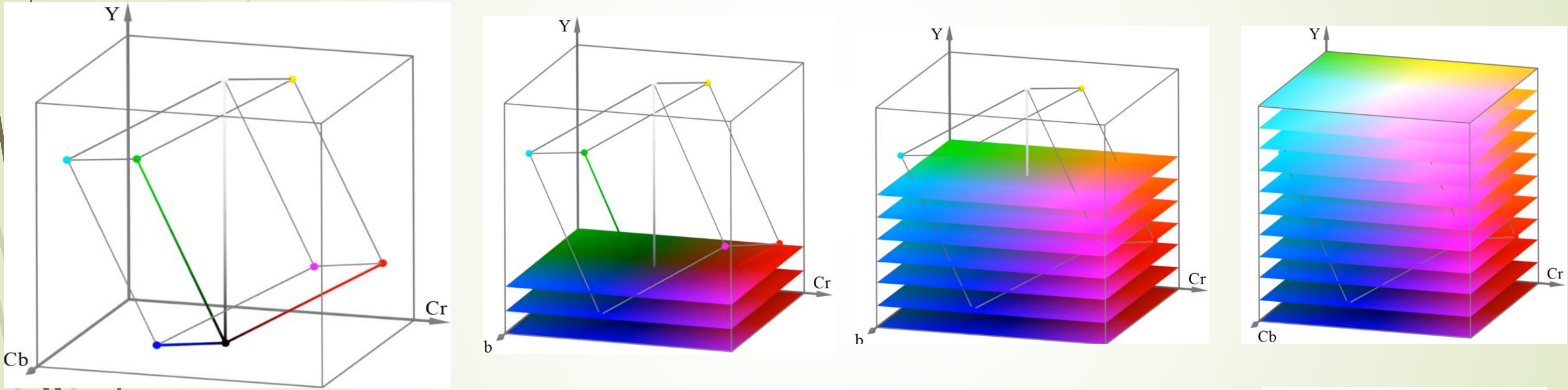
Y = 0.5



The center of the cube face is ***white***, and the center of the opposite side is ***black***. The line linking the both sides is the luminance component (from black to white). Therefore, each color can be represented by the center of one thin slice from the cube (luminance) + coordinates in the plane of ***Cb*** **/** ***Cr*****.**

Color Spaces (YCbCr)

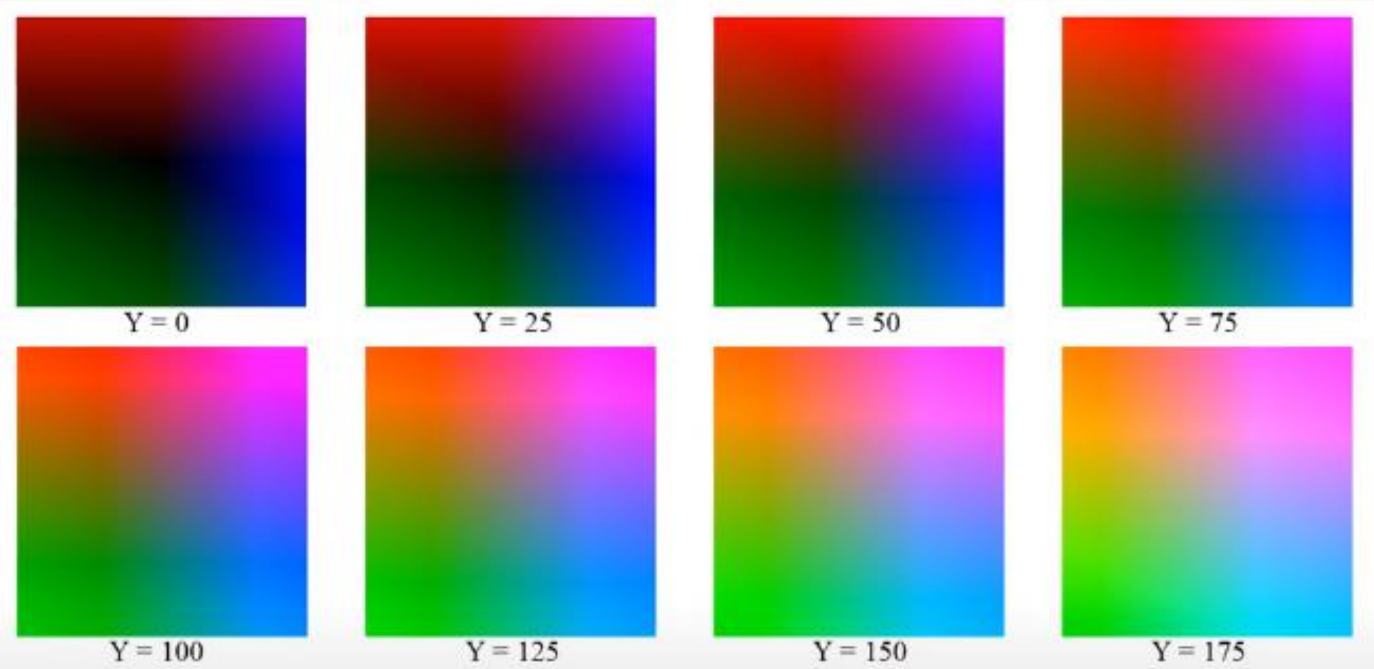
The RGB color space inside the YCbCr, here is shown how cube slices are constructed



We can notice the change in CbCr planes
with the intersection with the RGB cube

Color Spaces (YCbCr)

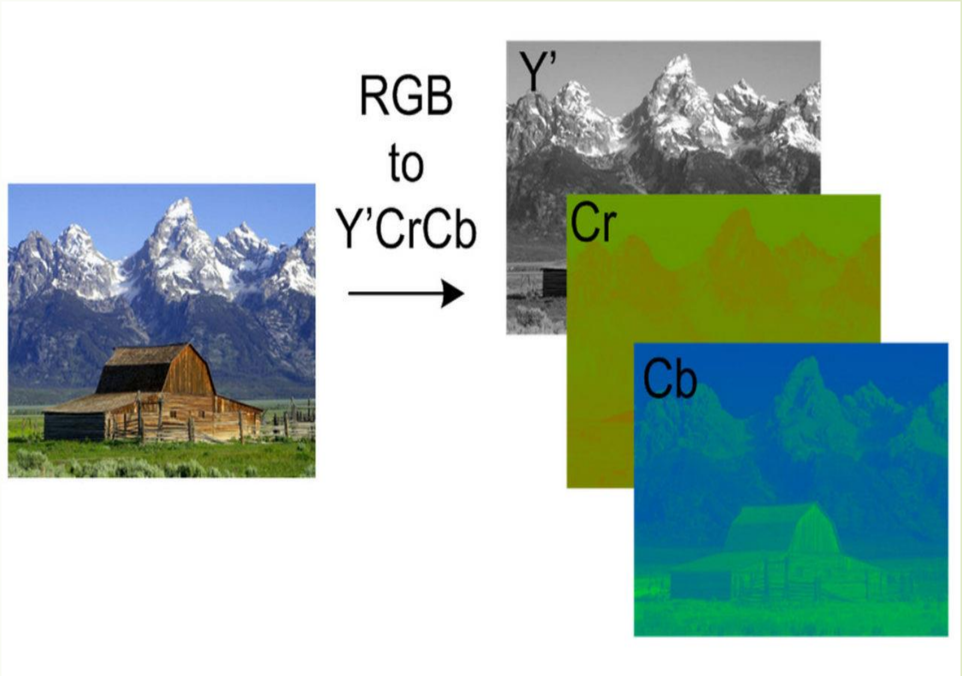
CbCr planes for different values of Y



Color Spaces (YCbCr)

A highly desirable property of YCbCr

Researchers have shown that humans are more sensitive to different levels of brightness than it is to differences in color. Since the brightness component (Y) is separated from the color (chromatic) components, these components are subsampled e.g., instead of considering 255 level (i.e., 8 bits), we can consider 16 level (i.e., 4 bits). This is called chroma. subsampling and it allows reducing the amount of bit allocated to the image.



Color Spaces (YCbCr)

Conversion between RGB and YCbCr

$$
\begin{gathered}
Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\
Cb = 0.564 \cdot (B - Y) + 128 \\
Cr = 0.713 \cdot (R - Y) + 128
\end{gathered}
$$

$$
\begin{gathered}
R = Y + 1.403 \cdot (Cr - 128) \\
G = Y - 0.344 \cdot (Cb - 128) - 0.714 \cdot (Cr - 128) \\
B = Y + 1.773 \cdot (Cb - 128)
\end{gathered}
$$

Image Basics

- **Quantization Algorithm**

Algorithm: Quantization;

Input: image **I (N,M)**; int **GL_Values** [1..K]; int **Nbre_Levels**; Boolean **B**;
Output: image **QI**;

```
Begin
For i=1 to N
  For j=1 to M
    B = Faux;
    For t=1 to K
      if (GL_Value[t] >= I(i,j) )
        QI(i,j) = GL_Value[t];
         B = Vrai;
         break;
      end
    end
If (B == Faux)
QI(i,j) = GL_Value[end];  % or 255
End end end
End.
```

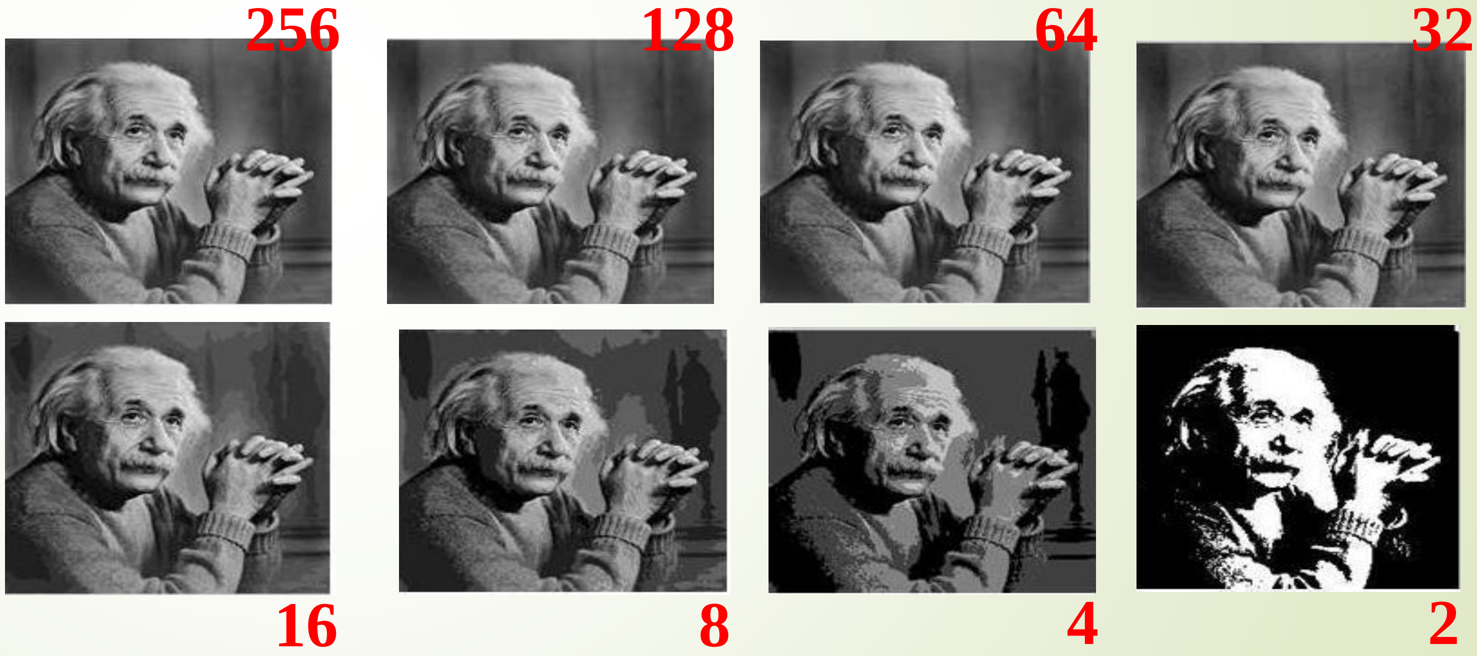
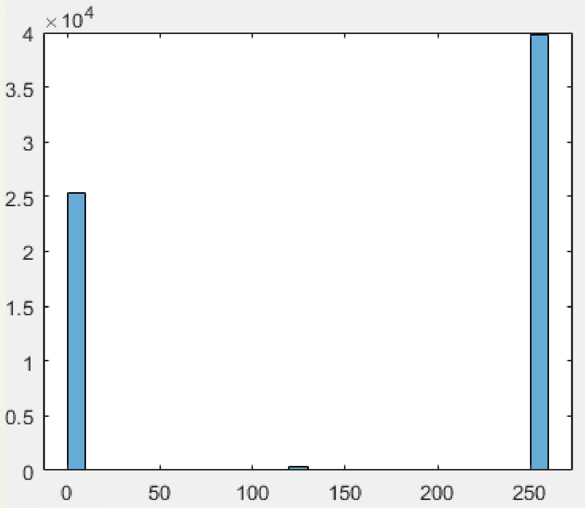


Image Basics

- **Morphological image processing**

Complementary image: replace the values of 0 by 255 and 255 to 0 in the binary images.



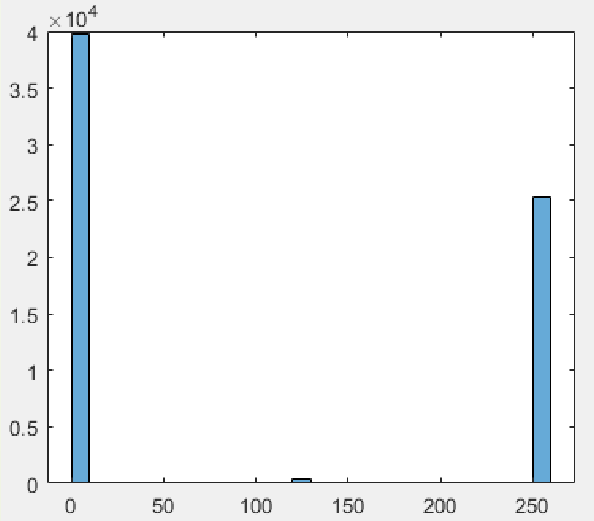
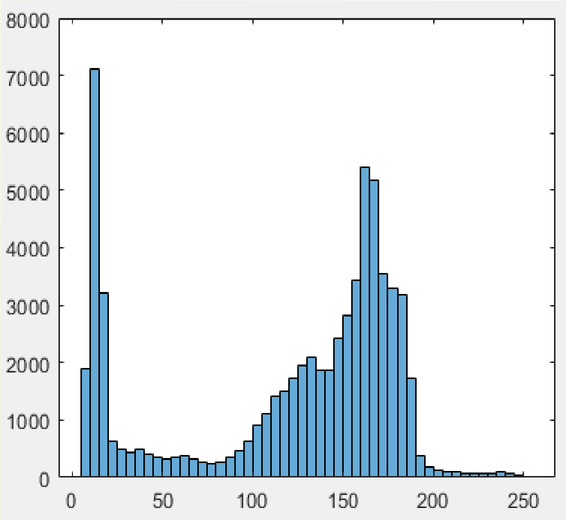


Image Basics

- **Morphological image processing**

Complementary image: the concept of complementarity can be generalized to gray-scale images where each value is replaced by its complementary to 255 i.e., X = 255 - X



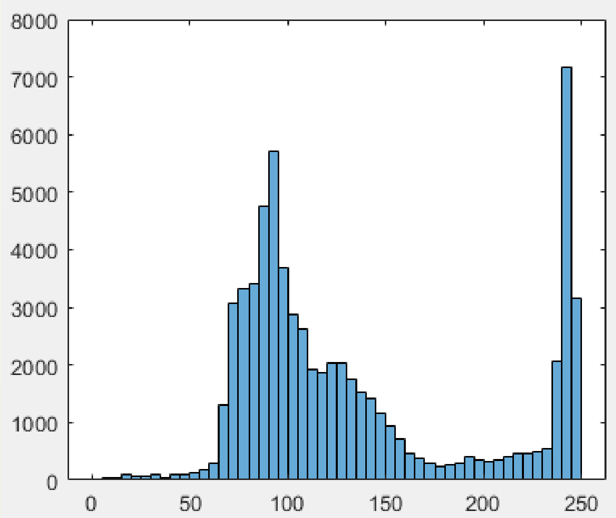
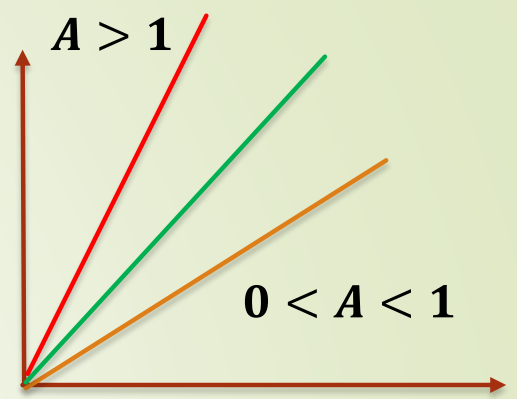
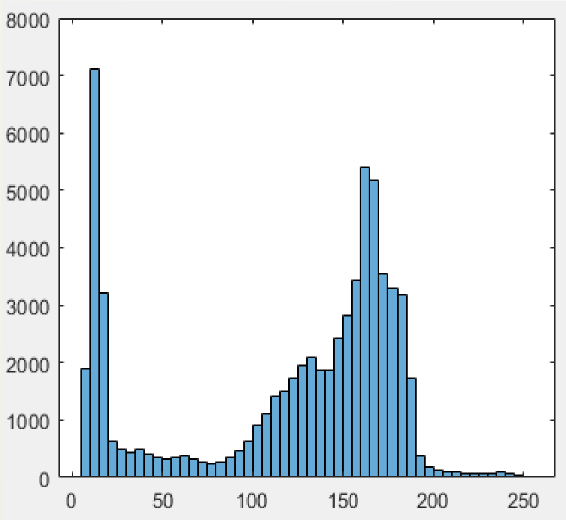


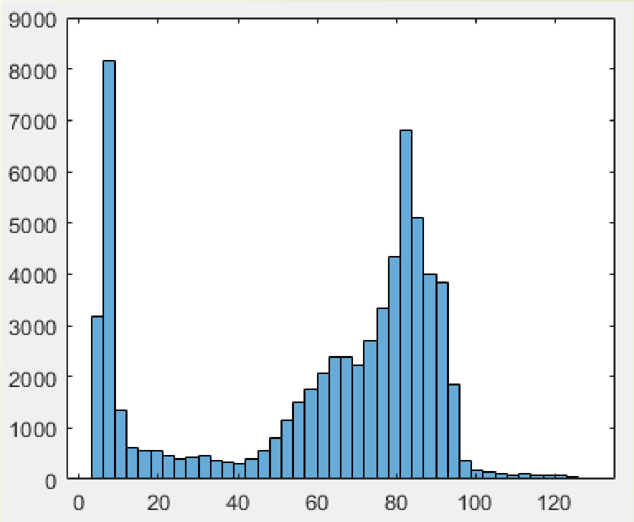
Image Basics

- **Morphological image processing**



Multiplication: which leads to increase / decrease the image brightness depending on the value By which the image values are multiplied $X = X \times A$. If $0 < A < 1$ then brightness will be Decreased, if A $>$ 1 brightness will increase.





$A = 0.5$

Image Basics

- **Morphological image processing**

Division: $X = \frac{X}{A}$. If $0 < A < 1$ then brightness will be increased, if A > 1 brightness will decrease.

$\boldsymbol{A = 0.5}$

$\boldsymbol{A = 2}$

Image Basics

- **Morphological image processing**

Max/ Min: takes the min / max from two different images.

MAX

MIN

Image Basics

- **Morphological image processing**

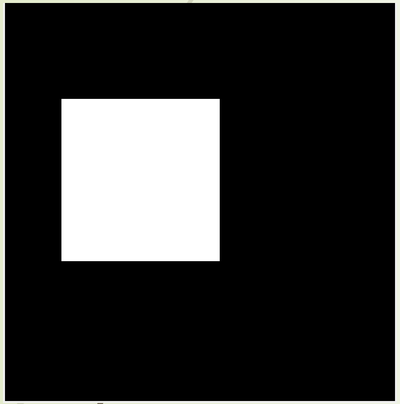
Logic operations: the previous operations were arithmetique operations, the logic operations include AND, OR, XOR, XNOR and other logic functions.

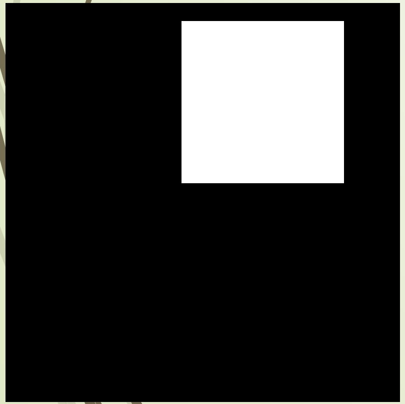
A	B	A AND B	A OR B	A XOR B	A XNOR B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1

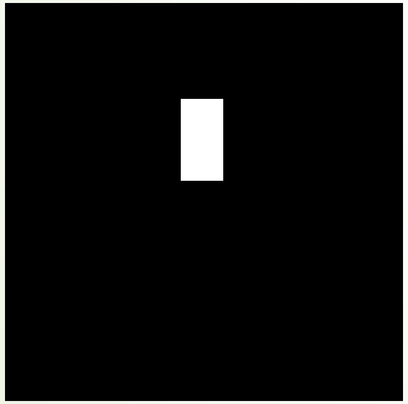
Image Basics

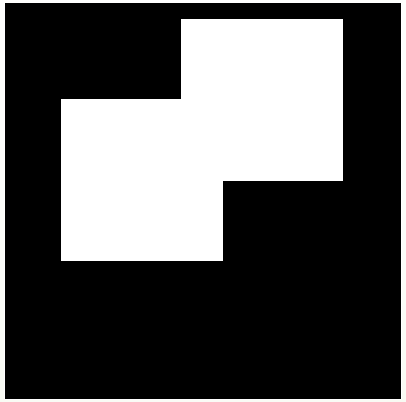
- ## Morphological image processing

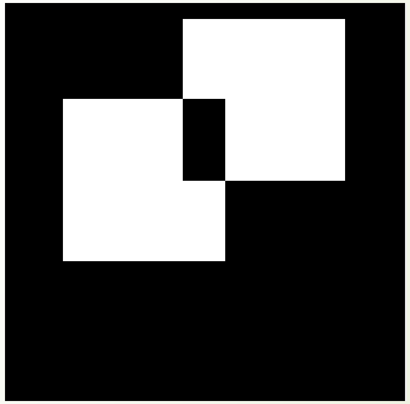
Logic operations: the previous operations were arithmetique operations, the logic operations include AND, OR, XOR, XNOR and other logic functions.

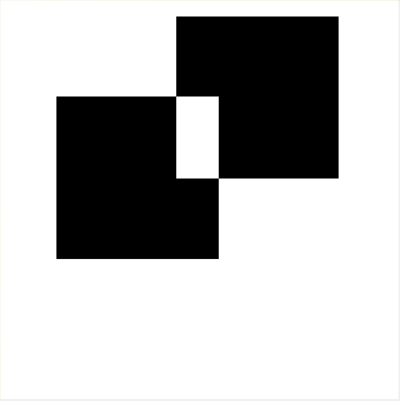












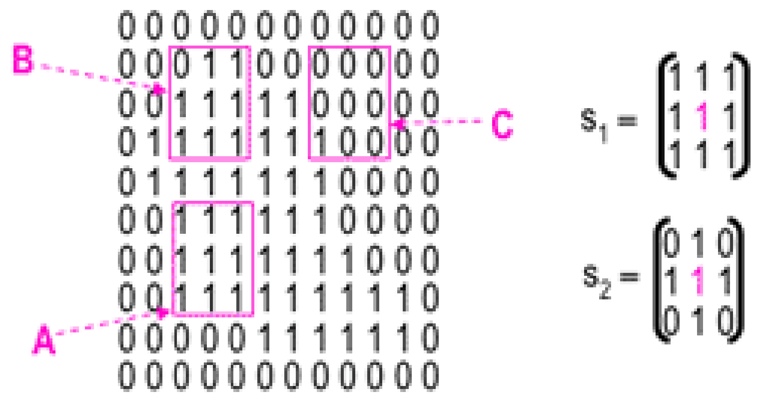
A AND B *A OR B* *A XOR B* *A XNOR B*

Image Basics

- **Morphological image processing**

Erosion and dilation:

The structuring element is said to ***fit*** the image if, for each of its pixels set to 1, the corresponding image pixel is also 1. Similarly, a structuring element is said to ***hit***, or intersect, an image if, at least for one of its pixels set to 1 the corresponding image pixel is also 1.

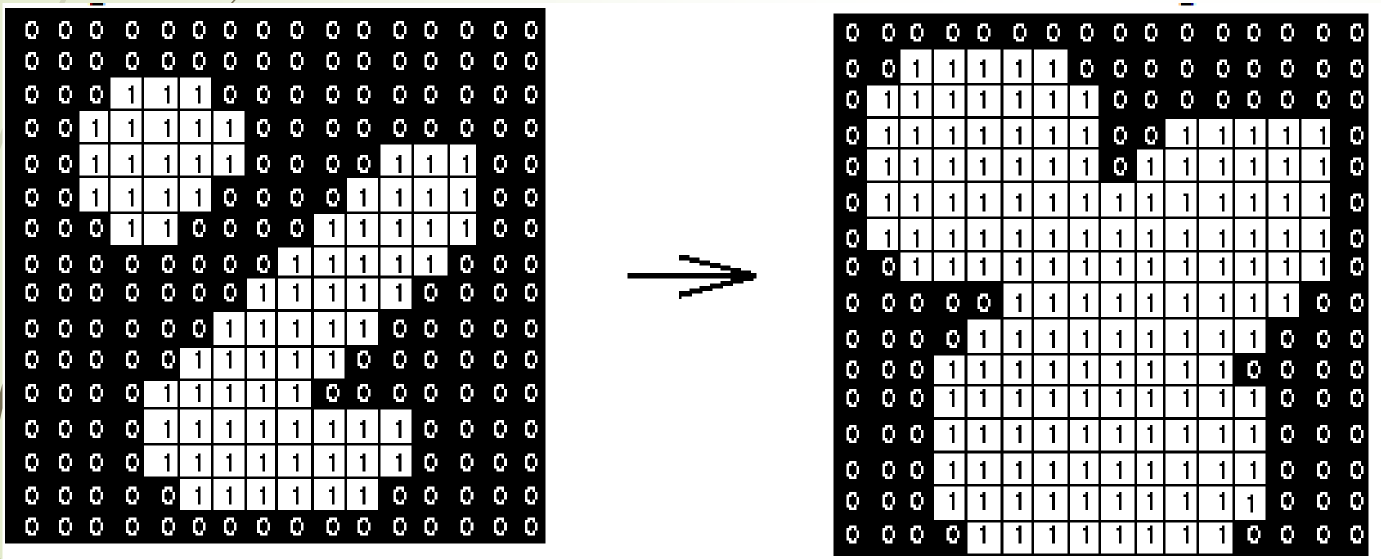


		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Image Basics

- ## Morphological image processing

Dilation of a binary image ***f*** with a structural element ***s*** denoted $f \oplus s$ yields a binary image $g = f \oplus s$ with ones in all locations (x, y) of structuring element's origin at which that element hits the input image i.e., $g(x, y) = 1$ is ***s*** hits ***f*** and **0** otherwise, repeating for all pixel coordinates (x, y).



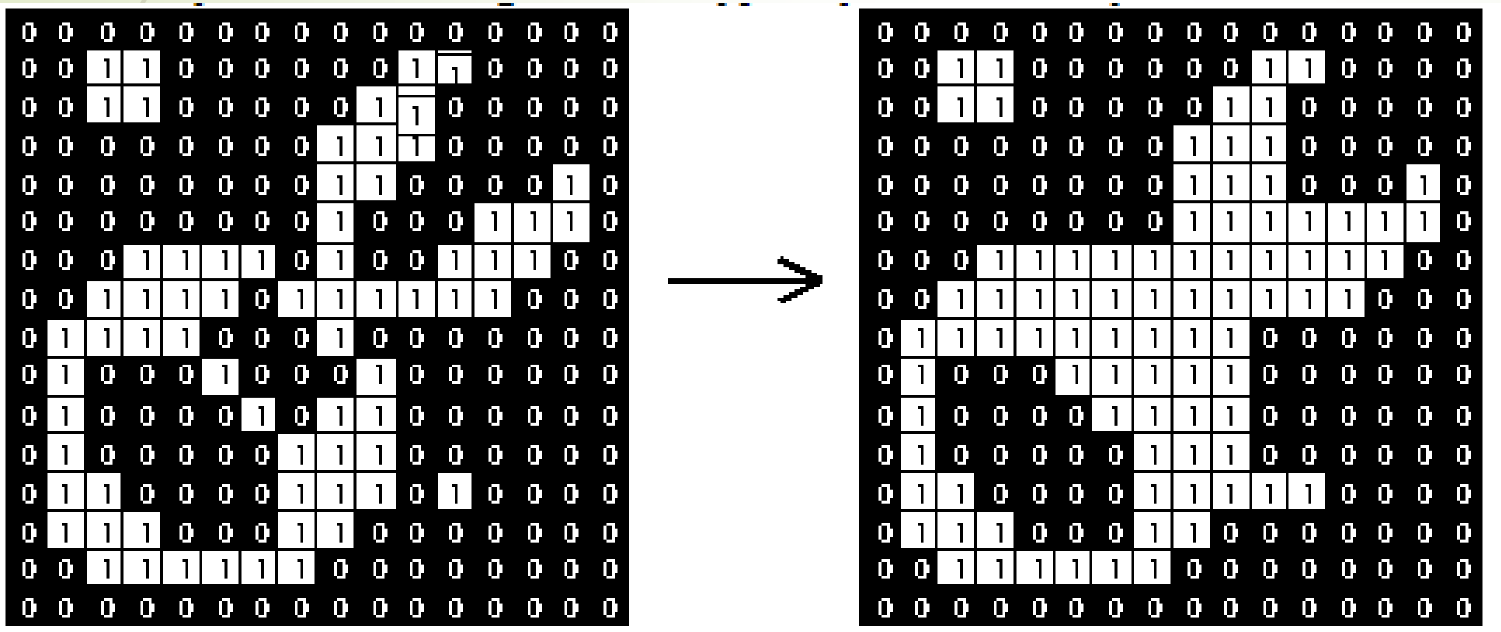
Dilation: a 3×3 square structuring element

The holes enclosed by a single region and gaps between different regions become smaller, and small intrusions into boundaries of a region are filled in

Image Basics

- **Morphological image processing**

Closing of a binary image ***f*** with a structural element ***s*** denoted $f \cdot s$ is a dilation followed by erosion $f \cdot s = (f \oplus s) \ominus s$

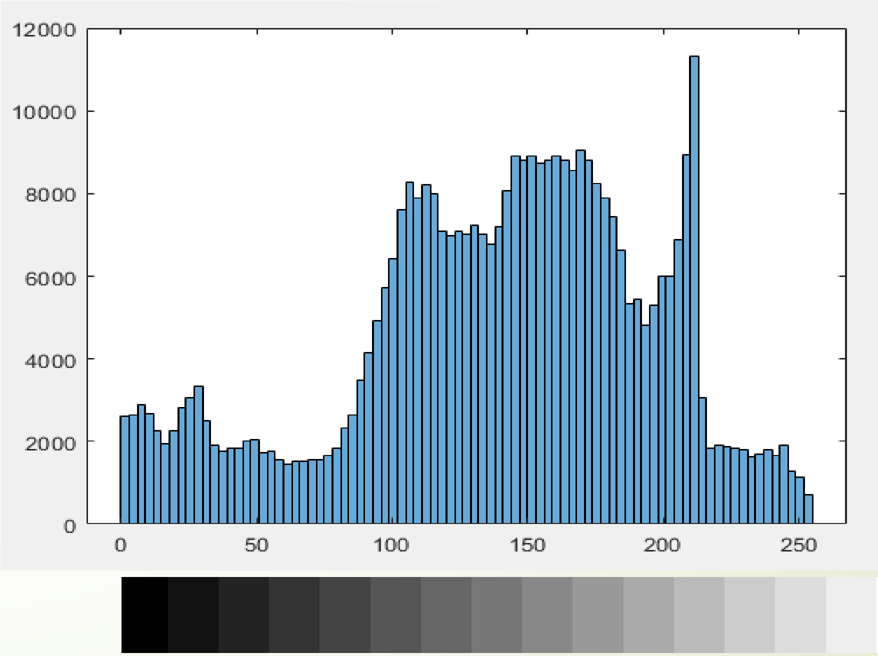


Closing with a 3×3 square structuring element

Closing is so called because it can fill holes in the regions while keeping the initial region sizes.

Improving Image quality

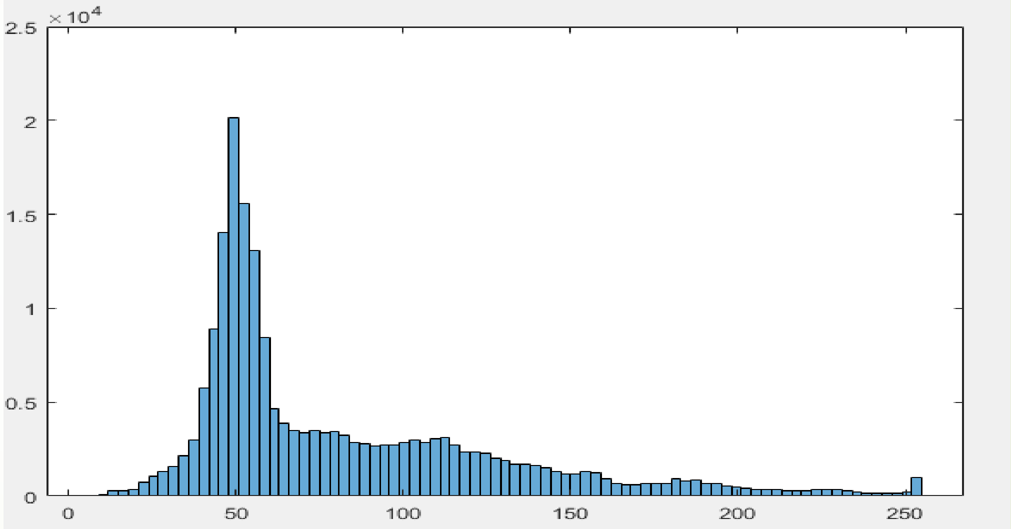
- **Gray-level image histogram**



On the y axis of this histogram are the frequency or count. And on the x axis, we have gray level values.

Improving Image quality

- **Improving image contrast using histogram sliding**



We can notice that frequent intensities are in the first half of the histogram (< 60). Thus, image tend to be dark. To get a brighter one, we perform histogram sliding.

Improving Image quality

- **Increase Brightness based on histogram sliding**

Algorithm: Increase_Brightness;

Input: image **I (N,M)**; int Value;
Output: image **I2**;
Begin

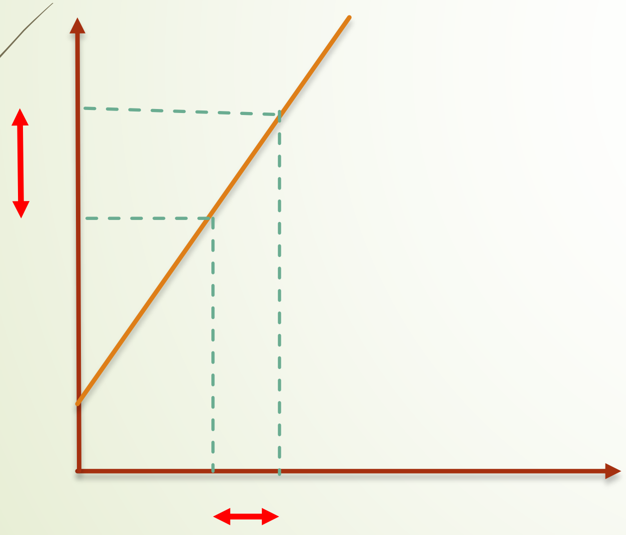
```
For i=1 to N
  For j=1 to M
        I2(i,j) = I(N,M) + Value;
  end
end
```

End.

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

Why $\boldsymbol{aX} + \boldsymbol{b}$



$\boldsymbol{a}$ is the slope and represents how steep the line is

We should select the $\boldsymbol{a}$ that results in a bigger interval

$\boldsymbol{b}$ is the y-intercept

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

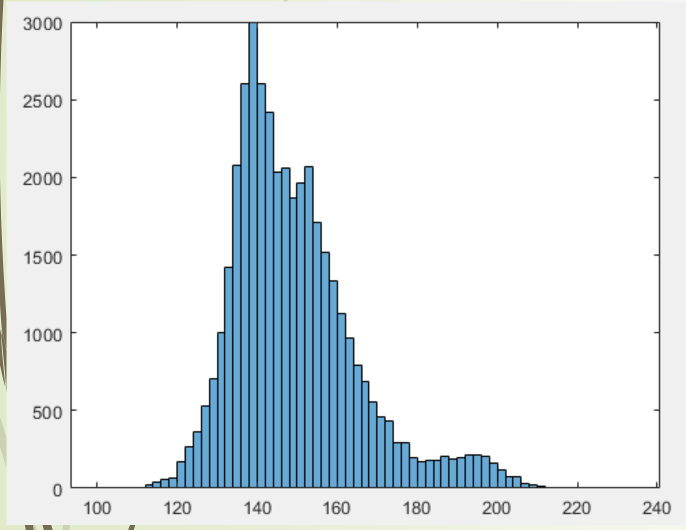
$$[I_{min} \;\; I_{max}] = [0 \;\; 185] \longrightarrow [R_{min} \;\; R_{max}] = [50 \;\; 255]$$

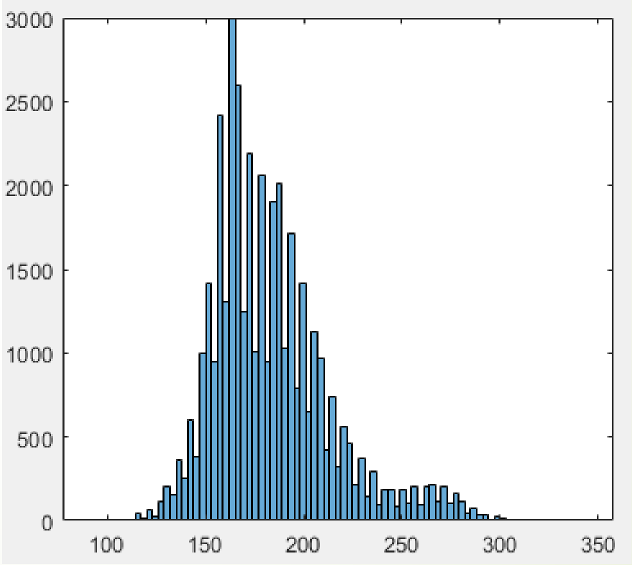
$$\boldsymbol{a} = \frac{\mathbf{255 - 50}}{\mathbf{185}} \qquad \boldsymbol{b} = \frac{\mathbf{185 \times 50 - 0 \times 255}}{\mathbf{255 - 50}}$$

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$$[I_{min}\ \ I_{max}] = [100\ \ 234] \longrightarrow [R_{min}\ \ R_{max}] = [0\ \ 255]$$





We can notice that the peaks are maintained

$$\boldsymbol{a} = \frac{\mathbf{255-0}}{\mathbf{234-100}} = \mathbf{1.9} \qquad \boldsymbol{b} = \frac{\mathbf{234\times0-100\times255}}{\mathbf{255-0}} = \mathbf{-100}$$

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

$$\boldsymbol{Dynamic\ range} = Max\ intensity\ - Min\ intensity$$

Algorithm: Adjust;

Input: image **I (N,M)**; % in the range $[I_{min}\ I_{max}]$
Output: image **R**; % in the range $[R_{min}\ R_{max}]$ int $\boldsymbol{R_{min}}$ $\boldsymbol{R_{max}}$;
Begin
For i=1 to N
  For j=1 to M
    R(i,j) = a * I(i,j) + b;
  end
end
End.

Improving Image quality

- **Adjust image dynamic to improve contrast (histogram stretching)**

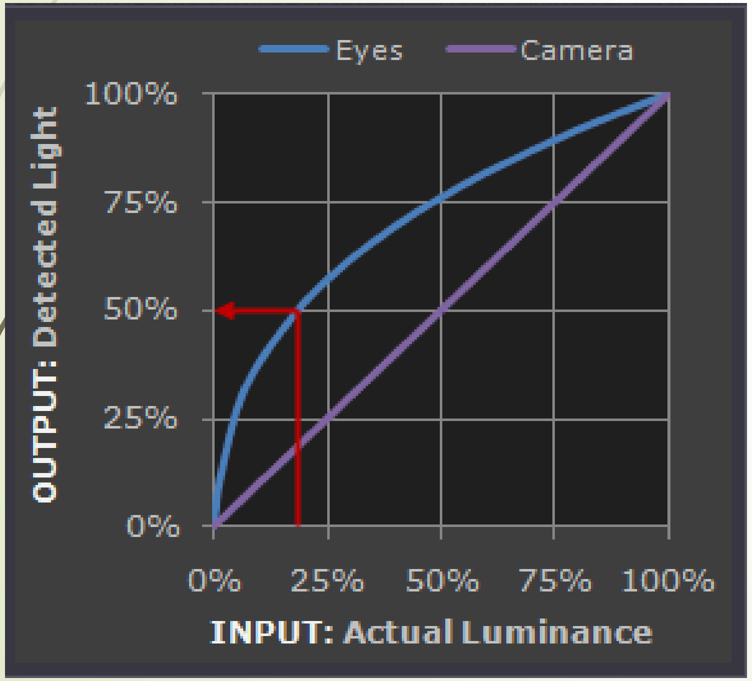
Another transformation can be expressed as

$$\boldsymbol{New} = \boldsymbol{R_{max}} \times \frac{\boldsymbol{X} - \boldsymbol{I_{min}}}{\boldsymbol{I_{max}} - \boldsymbol{I_{min}}}$$

The second term will be between 0 and 1. and $\boldsymbol{R_{max}}$ is the highest value of output image.

Improving Image quality

- **Adjust image dynamic to using Gamma correction**

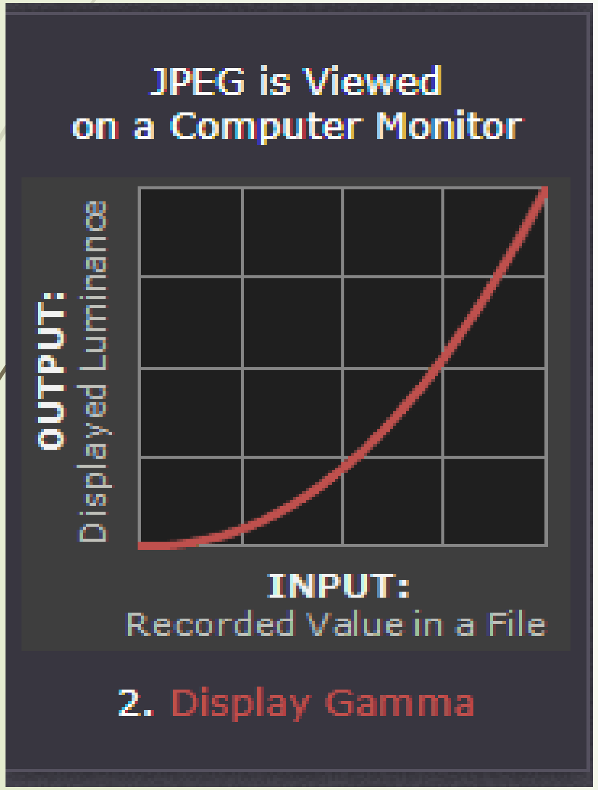


At first, we should know that our eyes don’t perceive light as the camera does. We can see that we are much more sensitive to dark levels than bright ones. For instance, for an actual light (luminance) of about **20%**, it is perceived brighter (**50%**).

For the camera, a linear equation is considered $aX + b = 0$. However, it is not the case for the human eyes.

Improving Image quality

- **Adjust image dynamic to using Gamma correction**



For a monitor, the light is viewed as in the figure (influence of video card and display device). The way the light is viewed by monitor can by expressed using the following equation

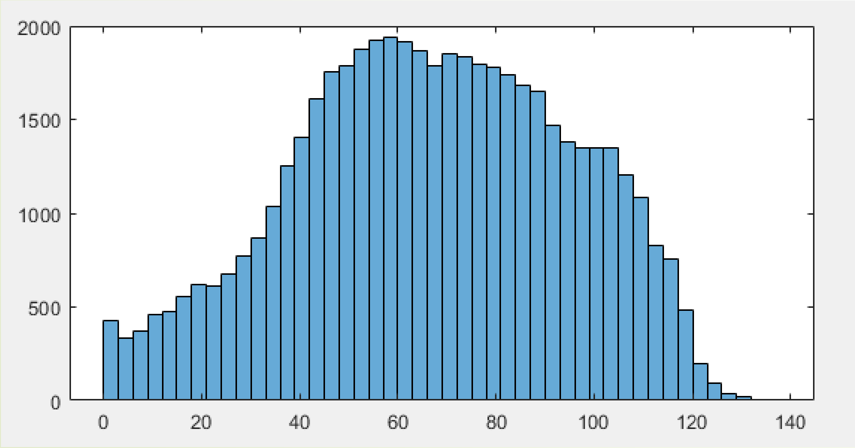
$$V_{out} = V_{in}^{\gamma}$$

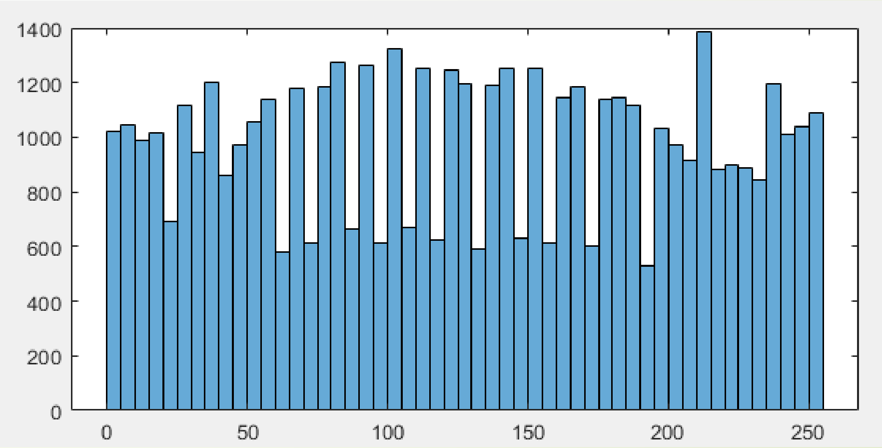
Often, $\gamma = 2{,}2$ (note that values are represented between 0 and 1, e.g., $0.5^{2.2} = 0.21$

Improving Image quality

- **Adjust image dynamic (improve contrast) by histogram equalization**

The aim of this process is to increase the image dynamic, as shown by the two figures below (0~130 become 0~250)





Improving Image quality

- **Gray-level image histogram**

It count the occurrence frequency of each gray-level in the image.

$$H(i) = \boldsymbol{Number\ of\ pixels\ having\ i\ as\ intensity}$$

Example

1	2	1	5
0	1	4	2
3	5	1	4
5	0	2	5

2	4	3	1	2	4

Improving Image quality

- Gray-level image histogram

Probability mass function (PMF)

1	2	1	5
0	1	4	2
3	5	1	4
5	0	2	5

2	4	3	1	2	4
2/16	4/16	3/16	1/16	2/16	4/16

Cumulative distribution function (CDF)

PMF	2/16	4/16	3/16	1/16	2/16	4/16
CDF	2/16	6/16	9/16	10/16	12/16	1

Improving Image quality

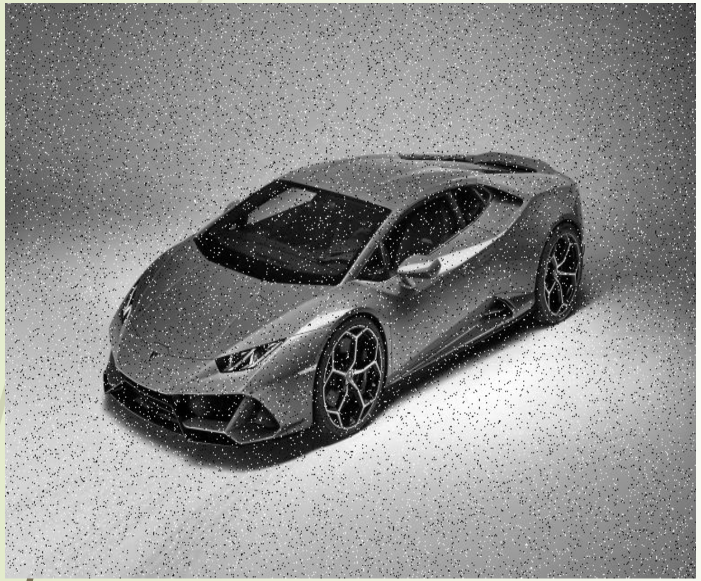
- **Adjust image dynamic (improve contrast) by histogram equalization**

Algorithm: Histeq;

Input: image **I (N,M)**; int L_{max};
Output: image **R(N,M)**; % improved image
Begin
$HC \; = CDF(I)$;
$Hist = Hist\ (I)$;
For i=1 to N
  For j=1 to M
    R(i,j) = $L_{max} \times HC(L_{i,j})$;
  end
end
End.

Improving Image quality

- **Removing noise**



Mainly there are several reasons behind the presence of noise in the image.

- Environmental factors which can negatively influence the imaging sensor.
- Low light and sensor temperature may cause image noise.
- Problems of quantification.
- Dust in the scanner can cause noise.
- Interference during transmission.

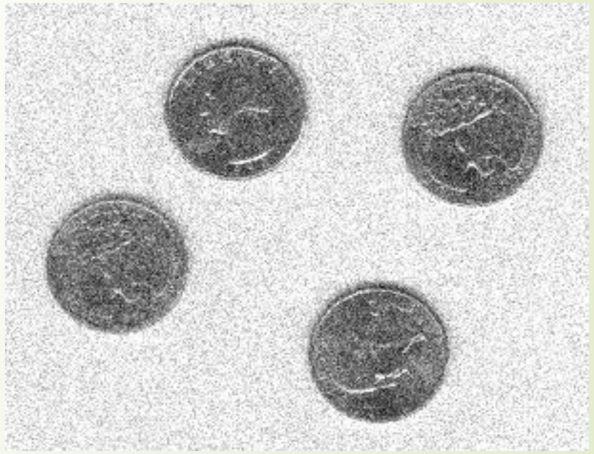
Improving Image quality

- **Removing noise**

There are several types of noise, among which we find the salt-pepper, and the Gaussian noise

Original image

Salt-pepper noise

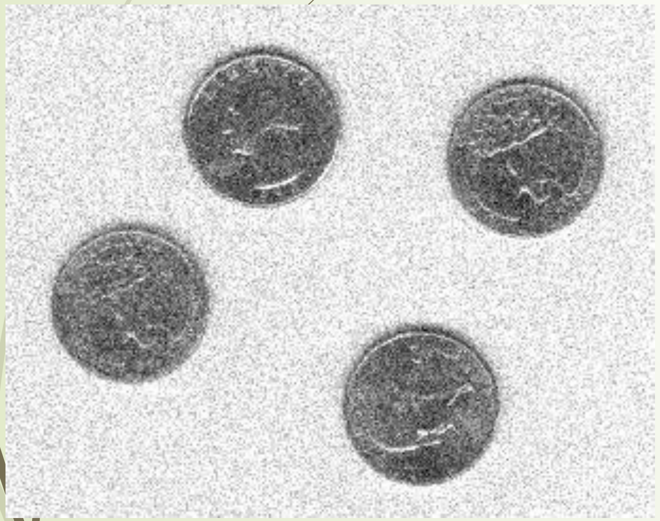


Gaussian noise

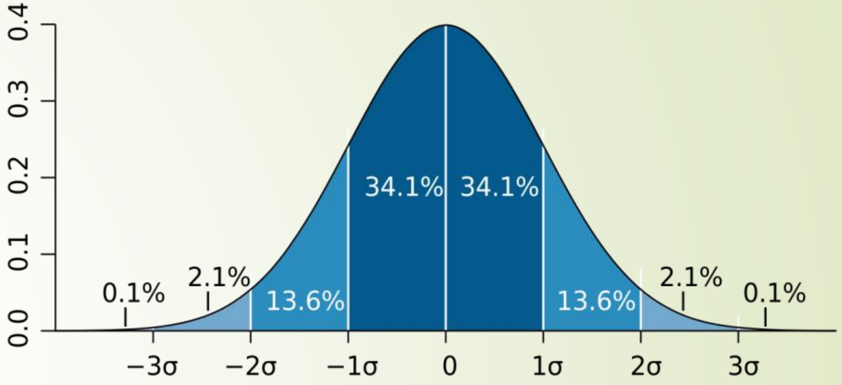
Improving Image quality

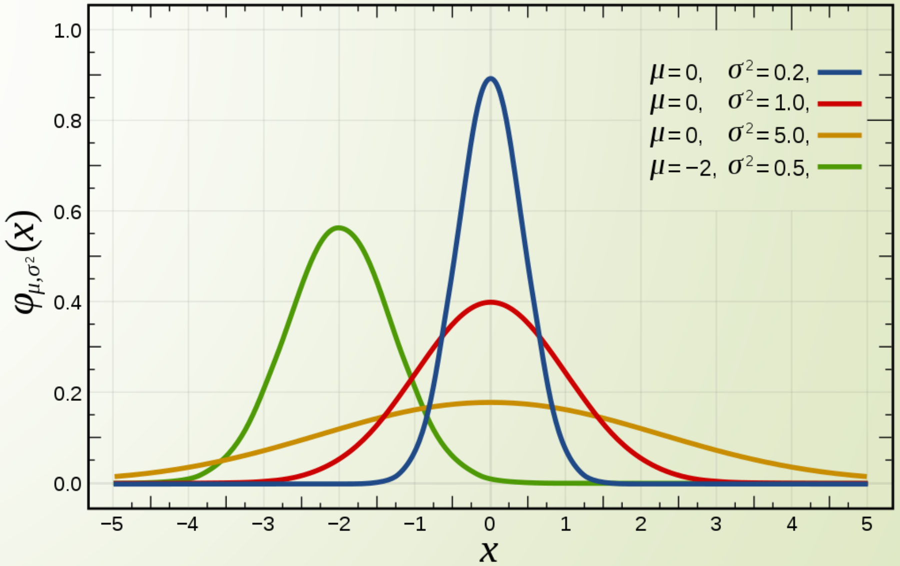
- **Removing noise**

Gaussian noise it is a statistical noise which is described by a probability density function. Random Gaussian function is added to the image function to generate this noise.



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





The previous types of noise are considered additive as they are added to the image $M = I + N$, other types of noise can be multiplicative.

Improving Image quality

- **Removing noise using mean filter**

One way to remove noise is by summing up the values of the Pixel neighborhood, divide them on the neighborhood size and put the result in the concerned pixel

2	1	0
5	20	3
2	6	4

2	1	0
5	4,77	3
2	6	4

$$\frac{(2+1+0+5+20+3+2+6+4)}{9} = 4,77$$

Improving Image quality

- **Removing noise using mean filter**

Original Image

Image smoothed with An average filter of 5x5

Image smoothed with An average filter of 11x11

We can observe that the blur increases as the neighborhood size increases. Much smoothing lead to loss image details.

Improving Image quality

- **Removing noise using mean filter**

The process of removing noise by browsing image top-down and left-right can be regarded as applying a filter on each image region. If we consider a neighborhood Of 3x3, the filter will looks like that

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	1	0
5	20	3
2	6	4

2	1	0
5	4,77	3
2	6	4

F * I = R

Applying a filter (called also kernel) on image is known as ***convolution***

Improving Image quality

- Removing noise using mean filter

F=

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Removing noise using mean filter**

F=

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

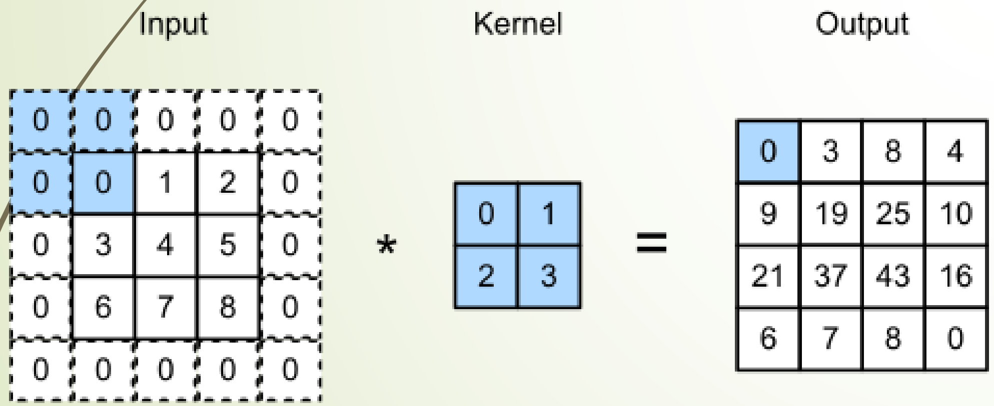
2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	2	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

- **Convolution versus cross-correlation**

Note that there is two different operations in this context namely ***convolution*** and ***cross-correlation***

Cross-correlation: measures the similarity between the patch and the image.



$$\boldsymbol{G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]\, F[i+u, j+v]}$$

Most explanations of convolution are actually presenting cross-correlation.

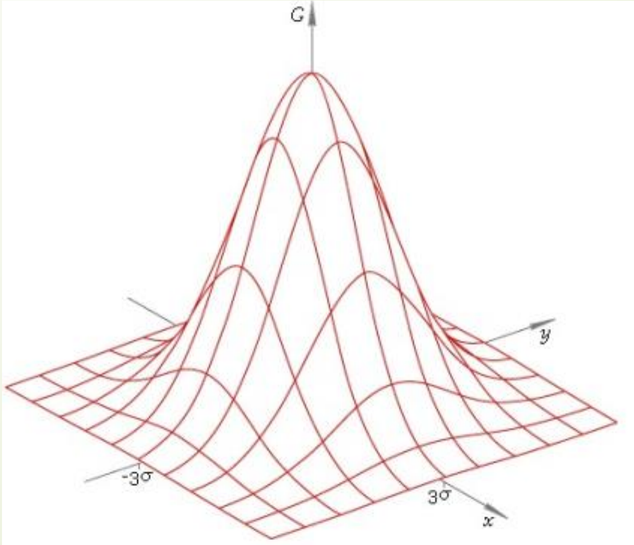
Improving Image quality

- **Removing noise using Gaussian smoothing**

In the Gaussian smoothing, near pixels to the concerned pixel contribute more than far ones when computing the new pixel value. To do so, we use the Gaussian function which is parameterized by the mean and standard deviation $\boldsymbol{\sigma}$. The 1D / 2D Gaussian is given by

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

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \times e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2}+\frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}$$



The centered and symmetric Gaussian is given by

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

$$\sigma_x = \sigma_y = \sigma$$

$$\mu_x = \mu_y = 0$$

Improving Image quality

- **Removing noise using Gaussian smoothing**

Algorithm: Noise_Removal;

Input: image **I (N,N)**; int **Filter_size**; double $\boldsymbol{\sigma}$;
Output: image **R(N,N)**;
Begin

Generate a filter $\boldsymbol{F}$ with $\boldsymbol{\sigma}$ as parameter
Convolve $\boldsymbol{I}$ with $\boldsymbol{I} * \boldsymbol{F} = \boldsymbol{R}$

End.

Improving Image quality

- **Separability of filters**

The mean filter can also be separated

$$\mathbf{1/9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow \mathbf{1/3} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \mathbf{1/3} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Improving Image quality

- **Removing noise using median filter**

Mean and Gaussian filters are linear filters as they can be expressed as $I \times F = N$. Where N is the resulting image F is the filter and I is the original image. However, linear filters are not well-suited for the salt-pepper noise. The median filter is a no-linear filter, which is well-suited to salt-pepper noise, and which preserves the image edges.

2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	3	6	1	5
0	3	1	51	2	1	4
8	1	0	0	8	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

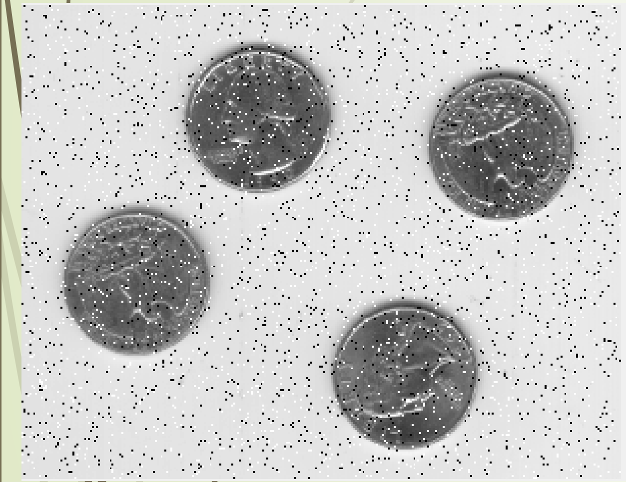
0	0	1	2	3	6	7	8	51

Median **Noise**

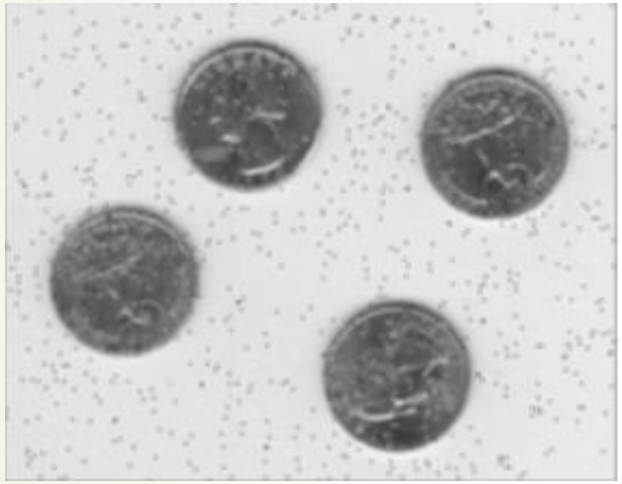
2	5	9	7	1	0	6
0	0	6	8	2	2	2
3	9	7	0	6	1	5
0	3	1	3	2	1	4
8	1	0	0	7	5	7
5	9	5	2	3	6	4
9	9	0	2	3	6	9

Improving Image quality

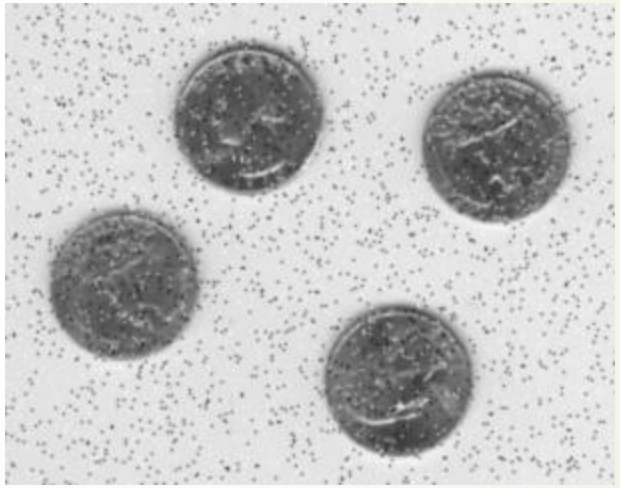
- **Removing noise using median filter**



Noisy image



Mean filter



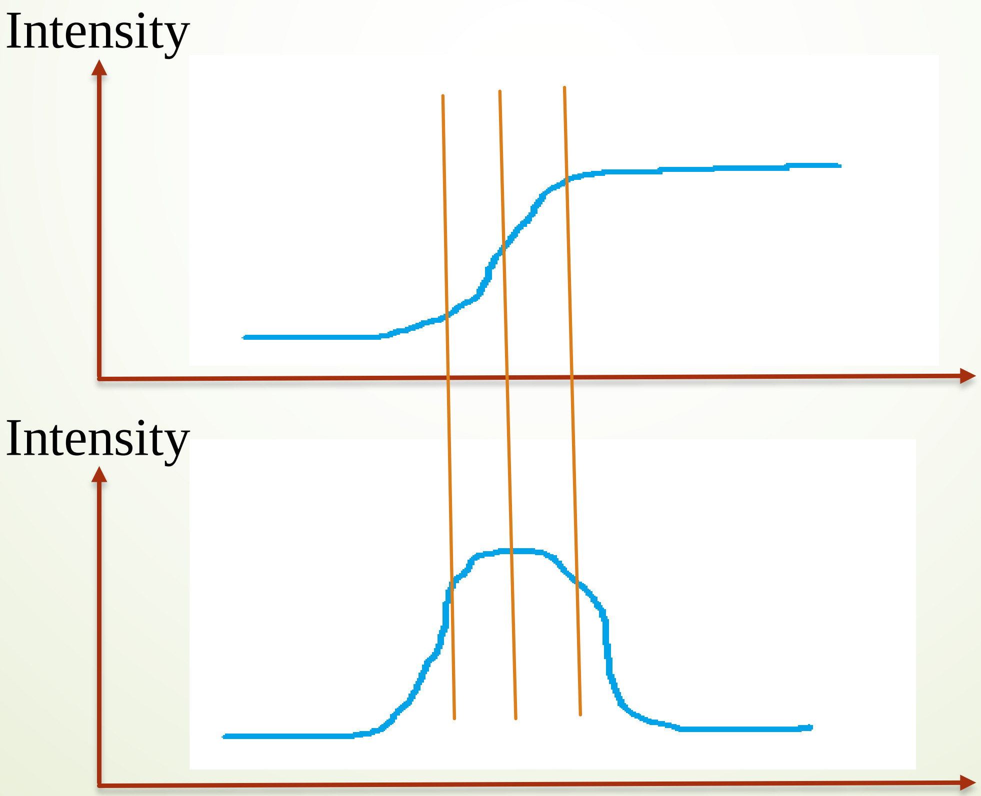
Gaussian filter

Median filter

Edge detection

- **How to detect edge**

Edge can be located by detecting the inflexion point, which is the maximum in the first derivative.



Edge detection

- **Sobel operator**

If we take the same image and we consider the horizontal detector we get the following

50	50	200	200
50	50	200	200
50	50	200	200
50	50	200	200

*

-1	-2	-1
0	0	0
1	2	1

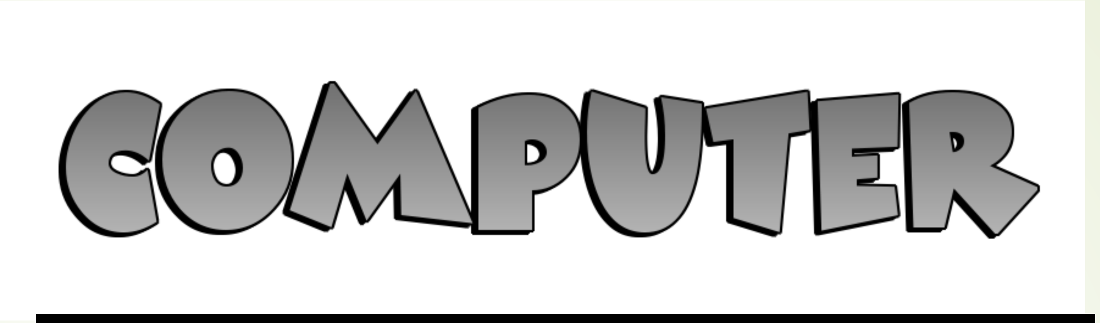
=

$$50 \times (-1) + 50 \times (0) + 50 \times (1) + 50 \times (-2) + 50 \times 0 + 50 \times 2 + 200 \times (-1) + 200 \times 0 + 200 \times (1) = 0$$

Here G_Y revealed that there is no horizontal edge in this image

Edge detection

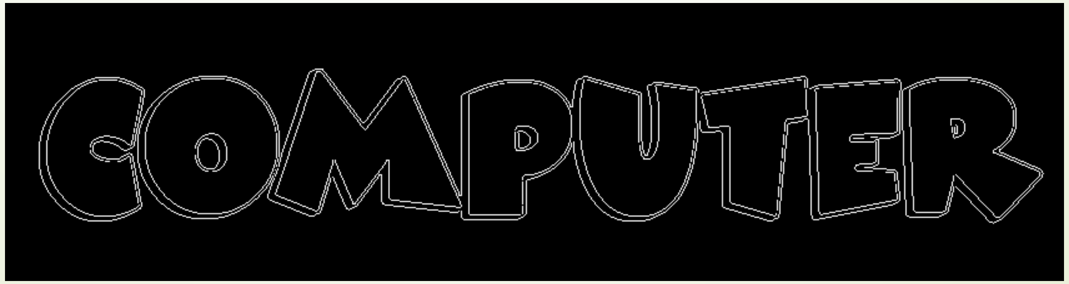
Image



Vertical edge

Horizontal edge

Final edge



Edge detection

Algorithm: Edge_detection;

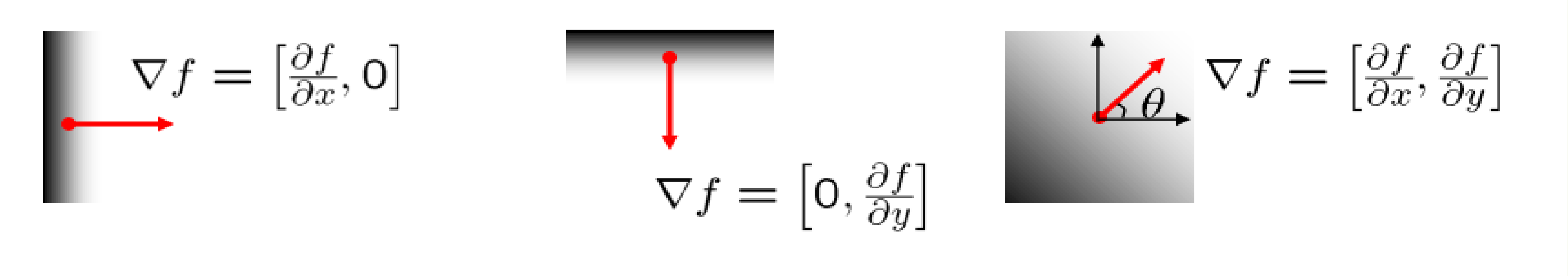
Input: image **I (N,N)**; Horizantal Filter **FH**; % example Sobel
Vertical Filter **FV**;
Output: image **R(N,N)**;
Begin

Compute the partial derevatives
- Convolve $\boldsymbol{I}$ with $\boldsymbol{I} * \boldsymbol{FV} = \boldsymbol{R_X}$
- Convolve $\boldsymbol{I}$ with $\boldsymbol{I} * \boldsymbol{FH} = \boldsymbol{R_Y}$

End.

Edge detection

- **Gradient magnitude and direction interpretation**

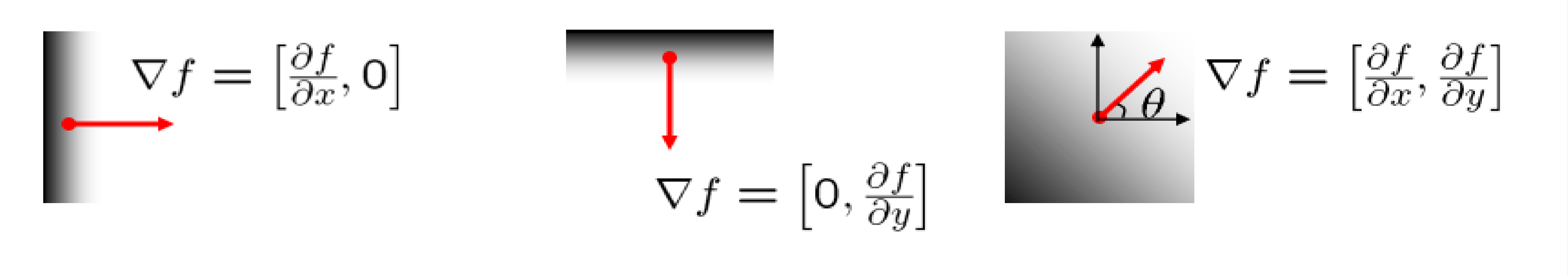


Compute gradient magnitude and orientation (for each pixel)

- Mag $(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})) = \sqrt{\boldsymbol{R}_X^2(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_Y^2(\boldsymbol{x},\boldsymbol{y})}$
- Orientation$\left(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})\right) = \text{atan}\left(\frac{R_X(x,y)}{R_Y(x,y)}\right)$

Edge detection

- **Gradient magnitude and direction interpretation**



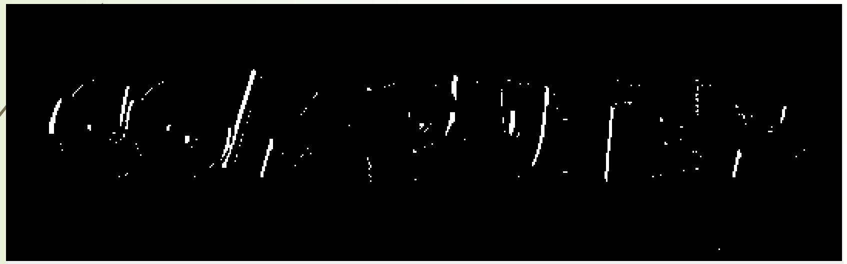
The gradient measure the change of image function. For instance, leftmost image change is only in X-axis, whereas, in the second image, change is in Y-axis. In the rightmost image, the gradient points in the direction of most rapid increase in intensity.

In brief, the magnitude of the gradient tells us how quickly the image is changing, while the direction of the gradient tells us the direction in which the image is changing.

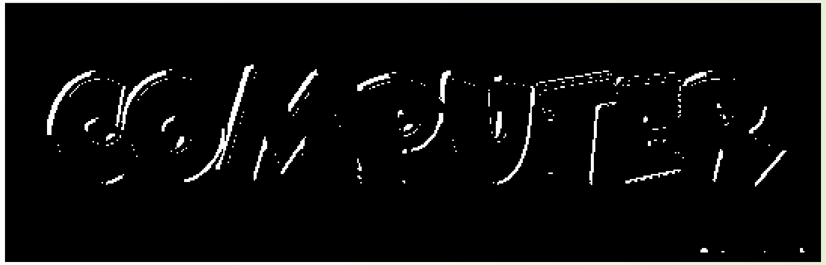
Edge detection
Using image gradient

The following images can be considered as edge images

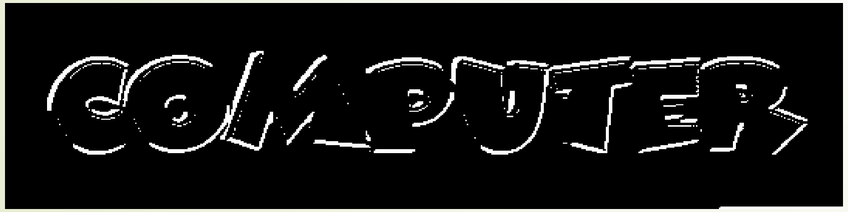
- Taking pixels having gradient magnitude greater than a certain threshold is also an edge image



Keep Magnitude > 150



Keep Magnitude > 100

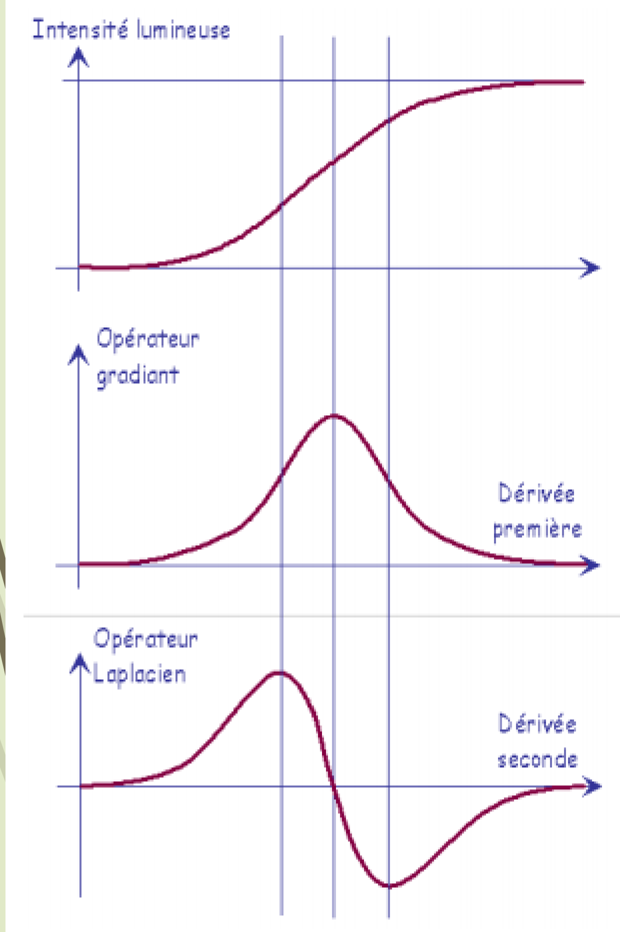


Keep Magnitude > 50

Keep Magnitude > 30

Edge detection
Using Laplacian of Gaussian

We can detect edge using the second derivatives of an image, then search for zero-crossings



Mathematically, second derivative is given by $\boldsymbol{I'(x) = I(x+1) - I(x)}$

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$
$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

This approximation is centered to (X+1) (note that h converges to 0, and we have considered *h=1*, so we subtract the 1 we added), now, if we replace X by (X-1), we get

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

Edge detection
Using Laplacian of Gaussian

However, this filter is sensitive to noise!

0	1	0
1	- 4	1
0	1	0

One way is to use Gaussian smoothing to alleviate noise before
Computing the image derivatives

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

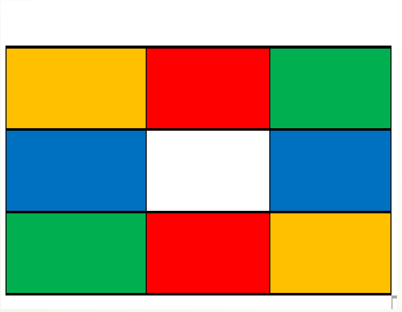
What do you think if we can simultaneously preform the both
Operations (i.e., derivation and smoothing), we can do so by
Using **Laplacian of Gaussian (LoG)**

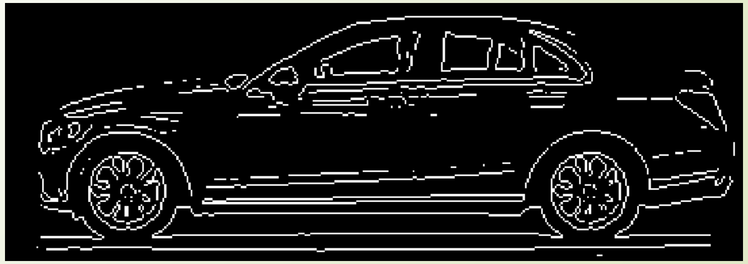
Edge detection
Using Laplacian of Gaussian

By convolving the input image using the Laplcian filter, we get another image. But how to detect edges! Edges = zero-crossings

zero-crossings: Using a 3x3 neighborhood centered at ***p***. A zero crossing at ***p*** implies that the signs of at least two of its opposing neighboring pixels must differ

- ❑ left/right
- ❑ Top/down or
- ❑ Diagonals





Edge detection
Using Laplacian of Gaussian

Algorithm: Edge_detection using LoG;

Input: image **I (N,N)**; **T:** threshold;
Output: image **R(N,N)**;
Begin

- $\boldsymbol{LoG_{XY}}$ = Generate the LoG filter using LoG function;
- Compute the second image derevative and smooth it
  - Convolve $\boldsymbol{I}$ with $\boldsymbol{I} * \boldsymbol{LoG_{XY}} = \boldsymbol{R}$
- Detect the zero-crossings in $\boldsymbol{R}$ (for each pixel)
- Keep only pixels for which $(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y}) > \boldsymbol{T})$

End.

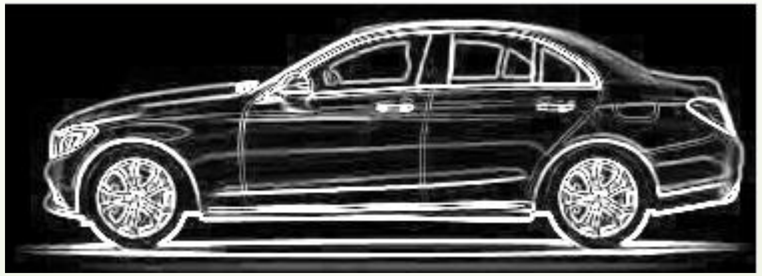
Edge detection
Using Canny algorithm

- The second step is to calculate the image gradient magnitude/direction. Magnitude can be calculated using Sobel operator

$$\text{Mag}\,(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})) = \sqrt{\boldsymbol{R}_{\boldsymbol{X}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{R}_{\boldsymbol{Y}}^{\boldsymbol{2}}(\boldsymbol{x},\boldsymbol{y})}$$

$$\text{Orientation}\big(\boldsymbol{R}(\boldsymbol{x},\boldsymbol{y})\big) = \text{atan}\left(\frac{R_X(x,y)}{R_Y(x,y)}\right)$$

Original Image



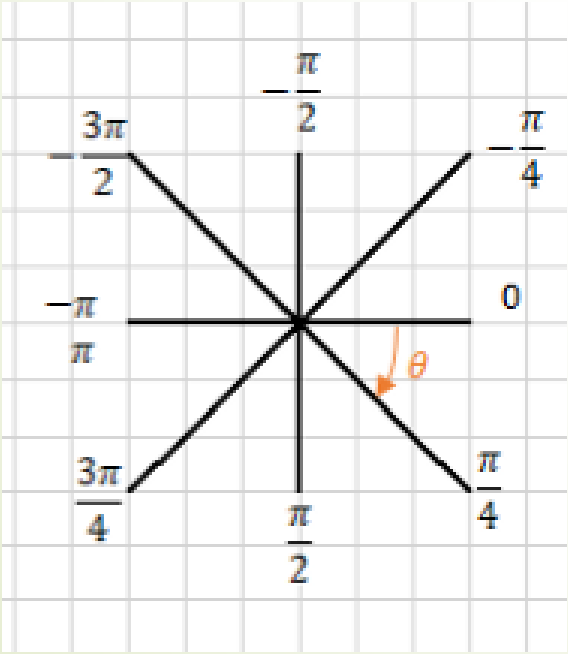
Gradient magnitude

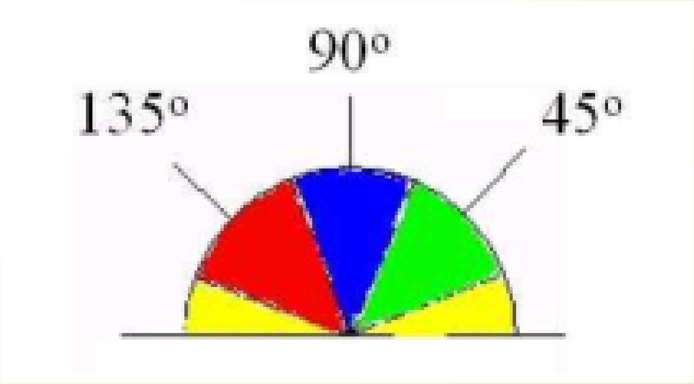
Gradient direction

Edge detection
Using Canny algorithm

- **Binning:** angles are rounded to 0°, 45°, 90° and 135°

Thus, $\boldsymbol{\theta} = 180°$ will be $\boldsymbol{\theta}' = 0°$ (bin 1), $\boldsymbol{\theta} = 225°$ will be $\boldsymbol{\theta}' = 45°$. If $\boldsymbol{\theta}$ is negative, thus add to it 180.



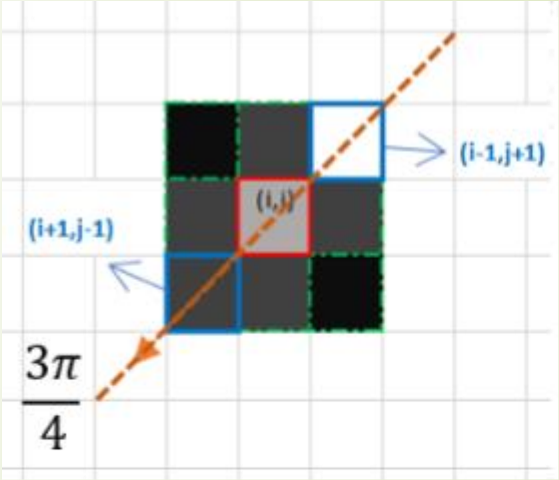


Edge detection
Using Canny algorithm

- **Non-Maximum Surpression**: keeps only those pixels on an edge with the highest gradient magnitude

So, three pixels in a 3×3 around pixel (x, y) are examined:

- If $\theta'(x, y) = 0°$, then the pixels $(x+1, y)$, (x, y), and $(x-1, y)$ are examined.
- If $\theta'(x, y) = 90°$, then the pixels $(x, y+1)$, (x, y), and $(x, y-1)$ are examined.
- If $\theta'(x, y) = 45°$, then the pixels $(x+1, y+1)$, (x, y), and $(x-1, y-1)$ are examined.
- If $\theta'(x, y) = 135°$, then the pixels $(x+1, y-1)$, (x, y), and $(x-1, y+1)$ are examined.



Here I(i-1, j-1) is kept

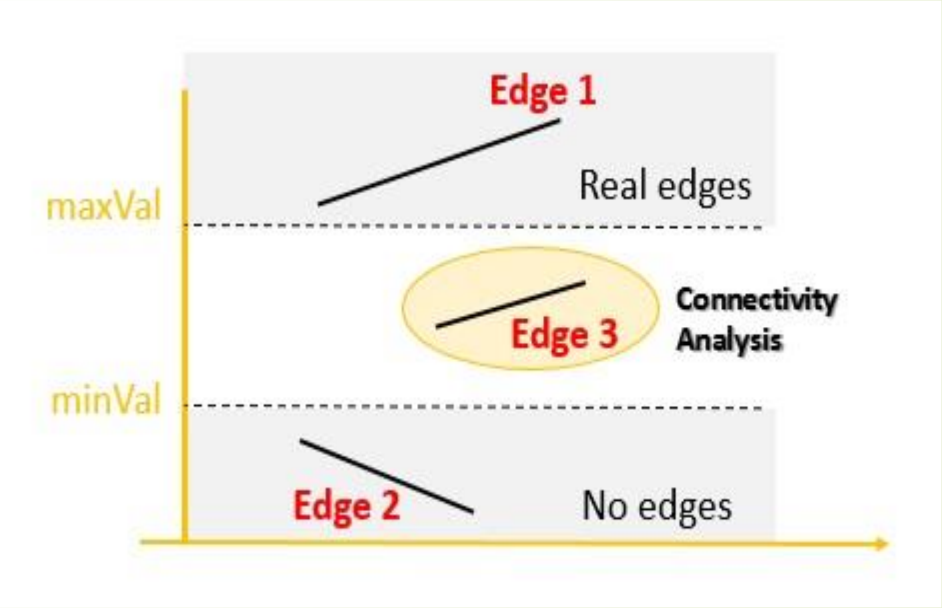
If pixel (x, y) has the highest gradient magnitude of the three pixels examined, it is kept as an edge. If one of the other two pixels has a higher gradient magnitude, then pixel (x, y) is not on the "center" of the edge and should not be classified as an edge pixel.

Edge detection
Using Canny algorithm

- **Hysteresis Thresholding**:

Some of the edges detected by Steps 1–3 will not actually be valid, but will just be noise. We would like to filter this noise out. Eliminating pixels whose gradient magnitude D falls below some threshold removes the worst of this problem, but it introduces a new problem.

A simple threshold may actually remove valid parts of a connected edge, leaving a disconnected final edge image. This happens in regions where the edge's gradient magnitude fluctuates between just above and just below the threshold. ***Hysteresis*** is one way of solving this problem. Instead of choosing a single threshold, two thresholds thigh and Tlow are used. Pixels with a gradient magnitude (D < Tlow) are discarded immediately. However, pixels with (Tlow ≤ D < Thigh) are only kept if they form a continuous edge line with pixels with high gradient magnitude (i.e., above Thigh).

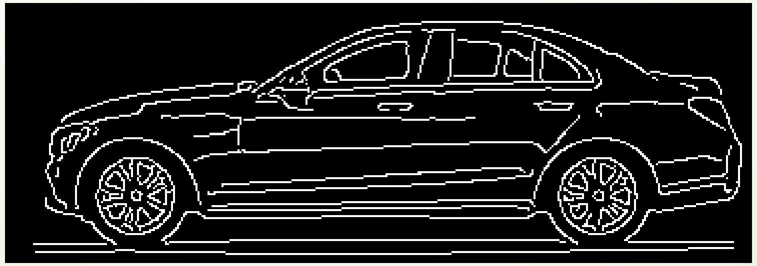


Edge detection
Using Canny algorithm

- **Hysteresis Thresholding**

- If pixel (x, y) has gradient magnitude less than t_{low} discard the edge (write out black).
- If pixel (x, y) has gradient magnitude greater than t_{high} keep the edge (write out white).
- If pixel (x, y) has gradient magnitude between t_{low} and t_{high} and any of its neighbors in a 3×3 region around it have gradient magnitudes greater than t_{high}, keep the edge (write out white).
- If none of pixel (x, y)'s neighbors have high gradient magnitudes but at least one falls between t_{low} and t_{high}, search the 5×5 region to see if any of these pixels have a magnitude greater than t_{high}. If so, keep the edge (write out white).
- Else, discard the edge (write out black).

Original Image



Edge detection using Canny