- Cones: Sensitive to colors. fine details.
- Rods: " to overall picture

Digital image:

$$f[x,y] = \begin{bmatrix} f[0,0], \cdots, f[W,0] \\ \vdots \qquad\qquad \vdots \\ f[0,H] \cdots f[W,H] \end{bmatrix},$$

$0 \le f[x,y] \le G-1$ where G is power of 2.

- The total energy collected by the sensor (eye, camera...) is used in order to calculate the gray level/intensity corresponding to that sensor.

method of sampling is determined by the sensor arrangement used to generate the image. When an image is generated by a single sensing element combined with mechanical motion, as in Fig. 2.13, the output of the sensor is quantized in the manner described above. However, spatial sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data. Mechanical motion can be made very exact so, in principle, there is almost no limit as to how fine we can sample an image using this approach. In practice, limits on sampling accuracy are determined by other factors, such as the quality of the optical components of the system.

When a sensing strip is used for image acquisition, the number of sensors in the strip establishes the sampling limitations in one image direction. Mechanical motion in the other direction can be controlled more accurately, but it makes little sense to try to achieve sampling density in one direction that exceeds the sampling limits established by the number of sensors in the other. Quantization of the sensor outputs completes the process of generating a digital image.

When a sensing array is used for image acquisition, there is no motion and the number of sensors in the array establishes the limits of sampling in both directions. Quantization of the sensor outputs is as before. Figure 2.17 illustrates this concept. Figure 2.17(a) shows a continuous image projected onto the plane of an array sensor. Figure 2.17(b) shows the image after sampling and quantization. Clearly, the quality of a digital image is determined to a large degree by the number of samples and discrete intensity levels used in sampling and quantization. However, as we show in Section 2.4.3, image content is also an important consideration in choosing these parameters.

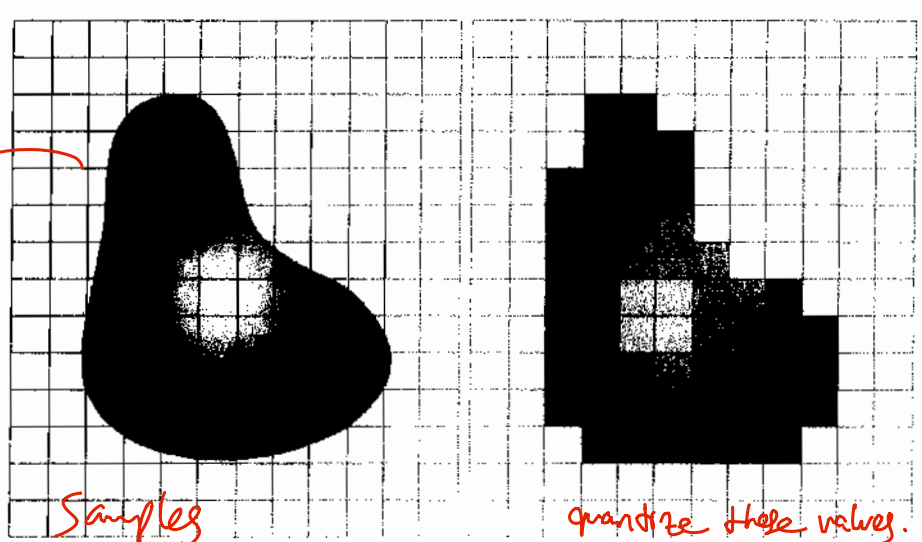


FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

How will represent the image with numbers?

Sampling

Quantization

∝ When we take digital picture, we're turning what we see in the real world into a format that computer can understand.

↳ Continuous image $f(x,y)$

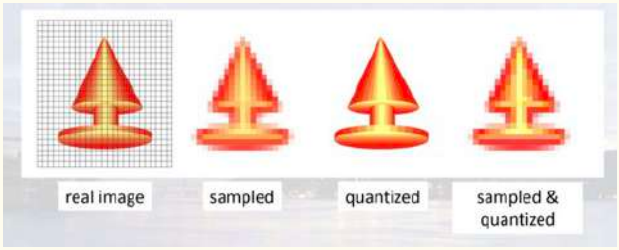
↳ discrete one $f[x,y]$

1-) Sampling : Dividing into tiny squares. → pixels

2-) Quantization : Giving brightness values to pixels.

Real Image → Sampled → quantized → sample ∝ quantized.

∝ Usually we use 8 bits to represent an image.



Shannon Sampling Theorem:

∝ Combination of signals

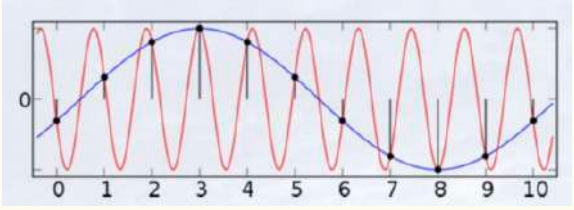
∝ When we want to turn this continuous wave into a digital form, we need to take snapshots. (regular samples)

∝ According to the theorem, we need to sample at least twice as fast as the highest frequency in the wave.

↳ to get more detail and not lose information.

∝ Moire effect happens when sampling frequency too low.

↳ when 2 similar but slightly different patterns overlap.



POINT PROCESSING :

$O[f(x,y)] = g(x,y)$ } Digital images are processed through "operators"

$a(x,y) = f(x,y) + g(x,y)$ ← represent the image as function

$s(x,y) = f(x,y) - g(x,y)$

⋮

Coordination of signals

In which situation we can do post processing?

1-) Noise averaging } Image come with noise bcs of the environment.

$f(x,y) = g(x,y) + n(x,y)$

g is noise free image

n uncorrelated noise

↓

taking the average of multiple images

Example: Noise Averaging with Multiple Images

Imagine taking three images of the same scene:

- Pixel at (x, y) in Image 1: 100 (due to noise)
- Pixel at (x, y) in Image 2: 110 (due to noise)
- Pixel at (x, y) in Image 3: 105 (due to noise)

If we average these pixel values, we get:

$$\text{Average} = \frac{100 + 110 + 105}{3} = 105$$

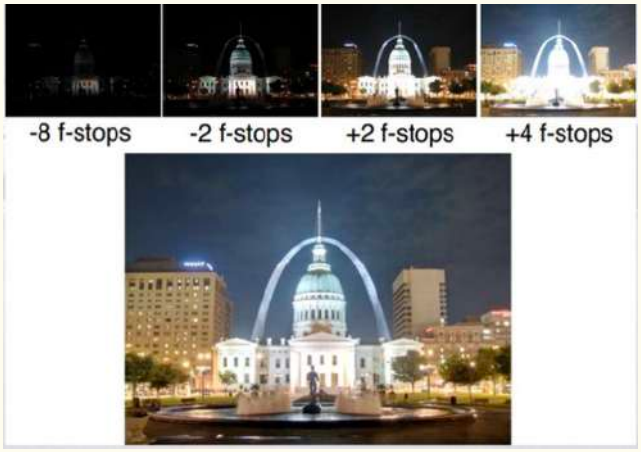
The result is a cleaner value for that pixel, closer to the true value without the noise effect.

2-) High Dynamic Range Imaging (HDRI)

- Another motivation to combining images.
- Exposure time long → you get more image from env.
- Over saturated } you will lose details.

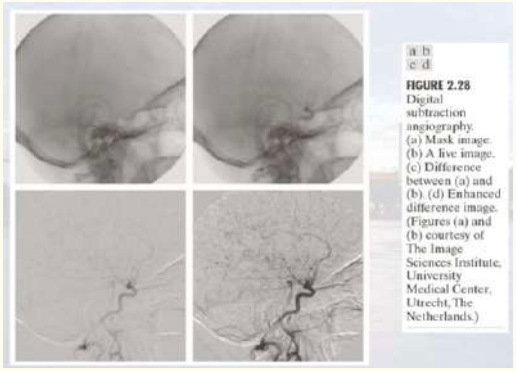
(high exposure time)

- HDR cameras capture several exposures of same scene with narrower ranges and combine them.



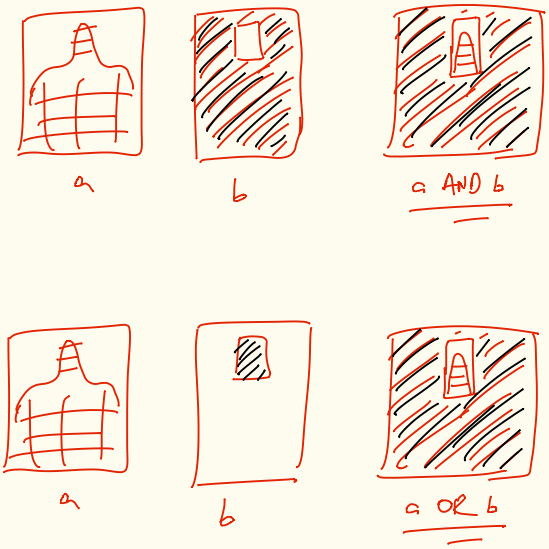
3-) Subtraction

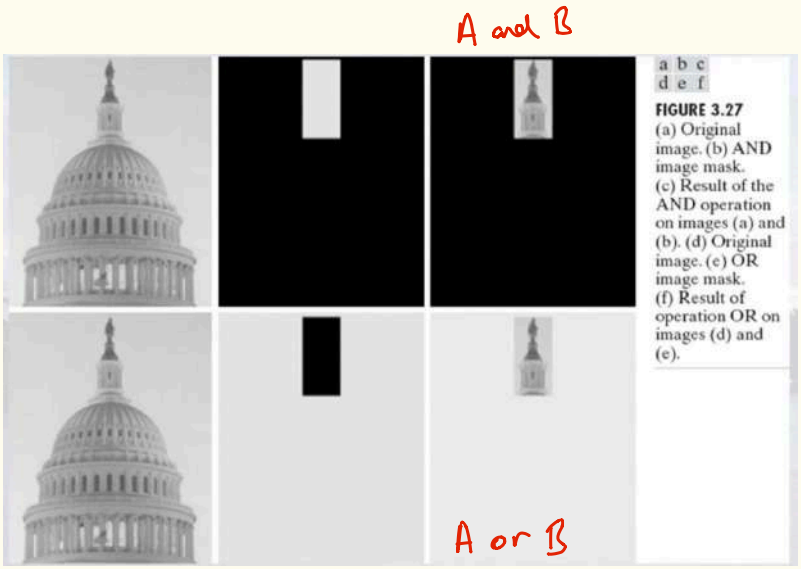
- can be used for highlighting differences.
- also serves for foreground detection.



a b
c d
FIGURE 2.28 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

Applying logical operations (AND, OR...)





a b c
d e f

FIGURE 3.27 (a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

3-) Intensity transformation.

- Gamma correction is type of intensity transformation.
- It adjusts the brightness of image } output = input$^{\gamma}$
- Gamma < 1, brighter
- Gamma > 1, darker.

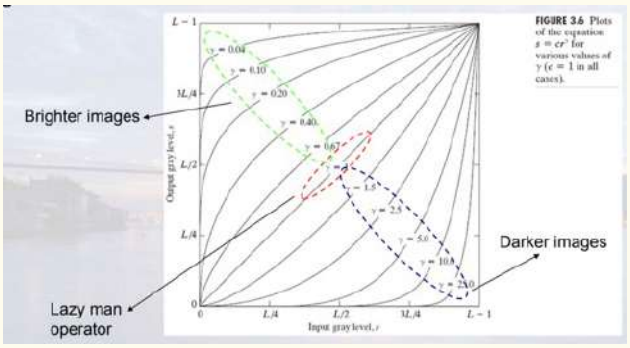


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

- **Zeroing Lower Order Bit Planes:** Results in loss of detail, reduced dynamic range, and a histogram that clusters around fewer high intensity levels, leading to posterization.
- **Zeroing Higher Order Bit Planes:** Leads to a drastic reduction in intensity range, clustering around low intensity values, loss of contrast, and severe posterization.

- Another intensity transformation } Bit-plane slicing
  - In a 8-bit image, each pixel has 8 bit. Bit plane slicing separates these 8 bits into separate images ("bit planes")

Noisy image — 0th ⟶ 7th — Brightest image

4-) Geometric transformations:

↳ includes 2 steps:

1-) transformation of spatial coordinates. (How pixels arranged in an image.)

2-) Intensity interpolation that assigns values as to the spatially transformed image.

$$(x,y) = T\{(v,w)\}$$

(x,y) → new location

(v,w) → pixel of original image

↳ $(x,y) = T\{(v,w)\} = (v/2, w/2)$

↓

Bayut düşürme (shrinks image into half of its size)

↳ We call these transformations as affine transformations.

2D image / $[x,y,1] = T \cdot [v,w,1] = [v,w,1] \cdot \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$

↓ 2 eksen

affine matrix

Transformation name	Affine matrix T	Coordinate equation
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = u$ $y = w$
Scaling	$\begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = C_x u$ $y = C_y w.$

$$\begin{bmatrix} s_x\ x \\ s_y\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = u \cdot \cos\theta - w \cdot \sin\theta$ $y = u \cdot \sin\theta + w \cdot \cos\theta$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \overbrace{\cos\theta & -\sin\theta}^{R} & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotation matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \qquad x = u + t_x \qquad y = v + t_y$$

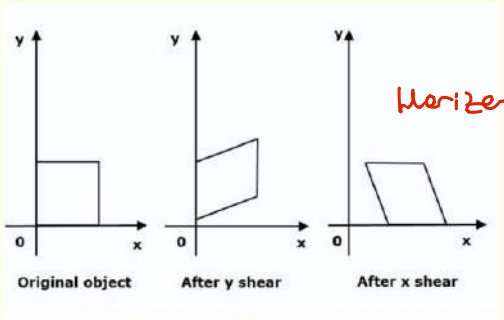
$$\Rightarrow P' \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear (vertical)

$$\begin{bmatrix} 1 & 0 & 0 \\ s_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad x = x' \qquad y =$$

" (horizontal)

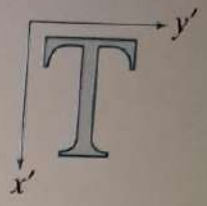
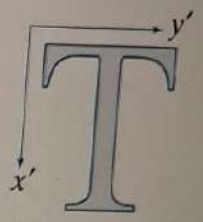
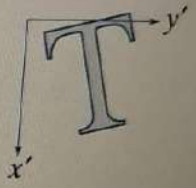
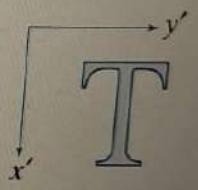
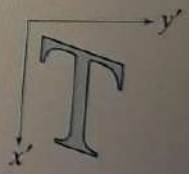
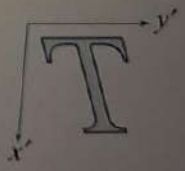
$$\begin{bmatrix} 1 & sh & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad x' = x + s_y \cdot y \qquad y' = y$$



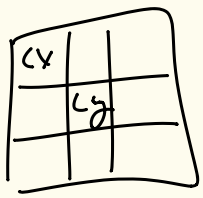
Horizontal.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotation matrix}} \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{translation matrix}} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

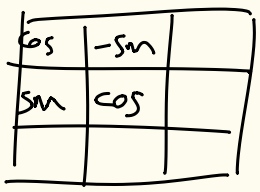
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R\,S & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to −1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

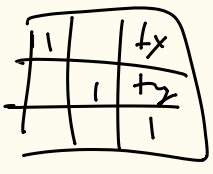
Scale



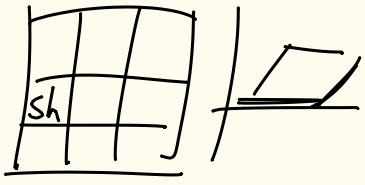
rotation



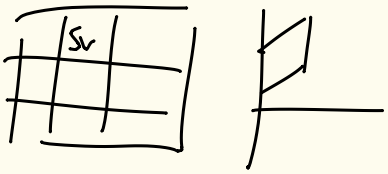
translation



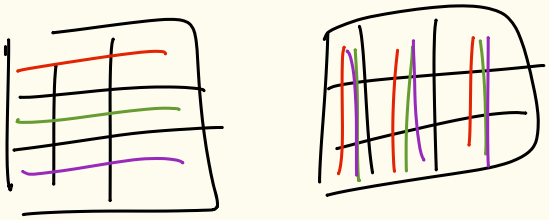
shear horizontal



shear vertical



Matriz Multr.



Inverse matrix

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

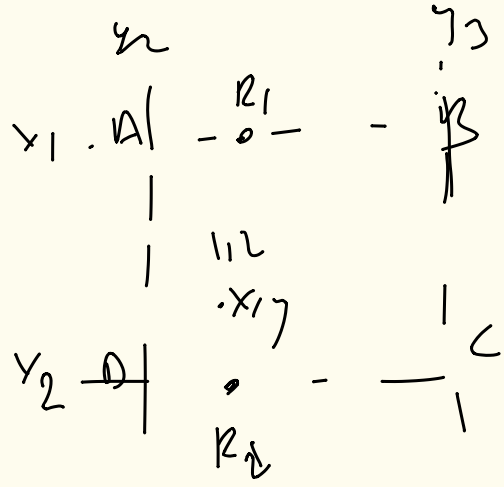
↓

$[A|I]$

↓

$$\left[\begin{array}{ccc|ccc} 1 & 2 & -1 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Interpolation



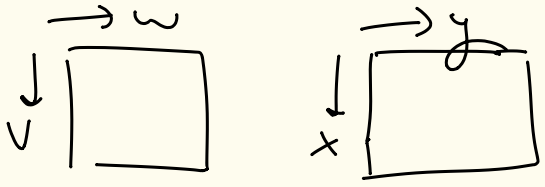
$$R_1 = \frac{y_3 - y}{y_3 - y_2} \cdot A$$

$$R_2 = \frac{y - y_2}{y_3 - y_2} \cdot B$$

$$P = R_1 \cdot \frac{x_2 - x}{x_2 - x_1} + R_2 \cdot \frac{x - x_1}{x_2 - x_1}$$

Translation at intensity values (2 ways)

1-) Forward mapping



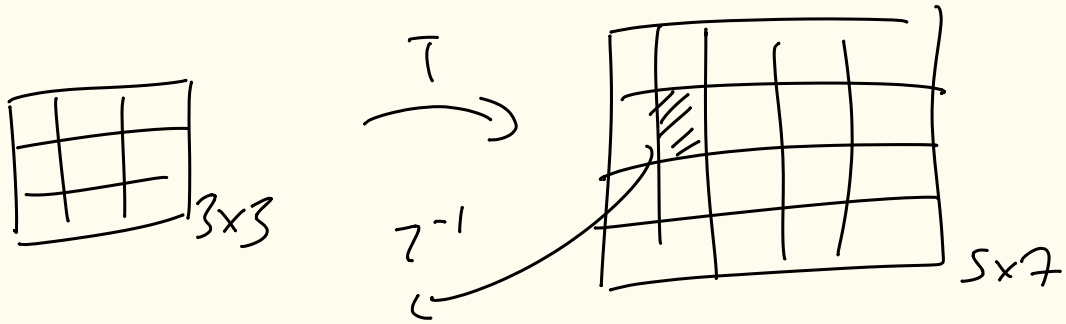
Scan pixels at the original image and compute the value at transformed location.

We can see some missing pixels at the output.

More than one pixel location in input image may project to the same location in output image.

2-) Inverse mapping

Scan the output image and for each pixel in output image find a pixel location in input image.



$$T^{-1}(2,2) = (1.3,\ 2.1)$$

forward mapping } new = matrix . orginal

Backward " } new = inverse . orginal
matrix

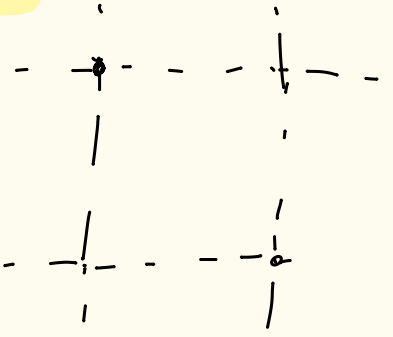
↳ Output image üzerinden, inverse matrix uygulayınca inputtaki yerini buluyoruz.

Daha sonra inputtaki pikseli, outputa boyuyoruz.

Bilinear interpolation:

$a = x - x_0$

$b = y - y_0$

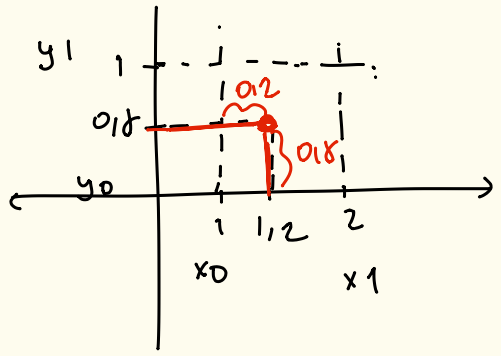


estimate this:

$(x, y) = (1.2, 0.8)$

$x0 = int(1.2) = 1 \qquad x1 = 2$

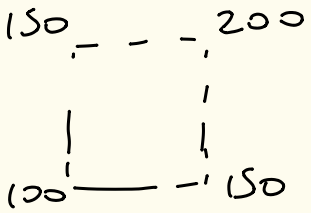
$y0 = int(0.8) = 0 \qquad y1 = 1$



$a = x - x0 = 1.2 - 1 = 0.2$

$b = y - y0 = 0.8 - 0 = 0.8$

Neighborhood pixels for example



pixel value

$(1-a).(1-b).top\_left + a.(1-b).top\_right \dots$

$0.8 . 0.2 . 150 + 0.2 . 0.2 . 200 \dots$

↓

to give more weight to the pixels that are closer.

∝ We need to make interpolation. (Filling the gaps.)

→ If you get 4 neighbors we call as "bilinear interpolation."

→ If you get 16 " " " "bicubic" ".

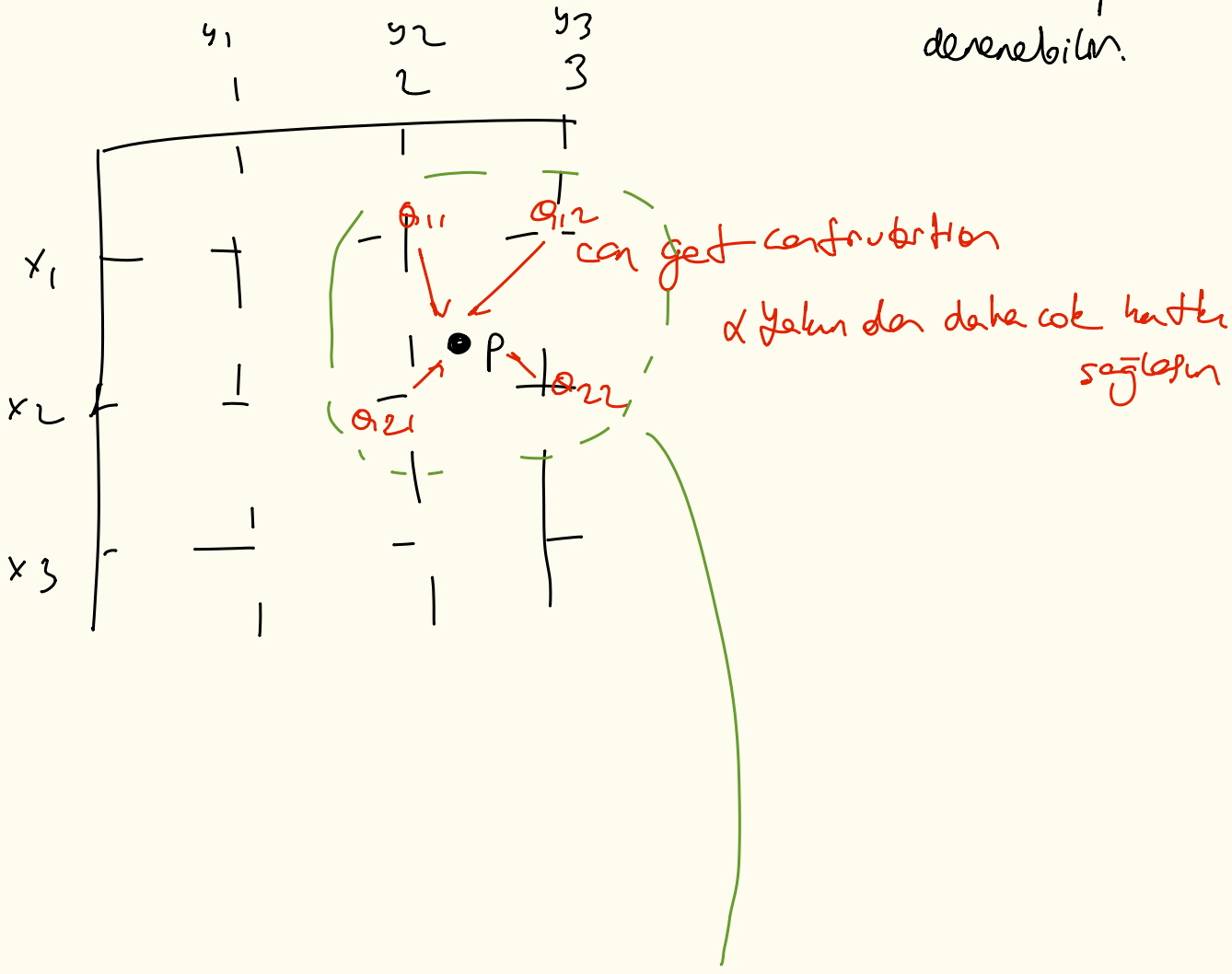
↓

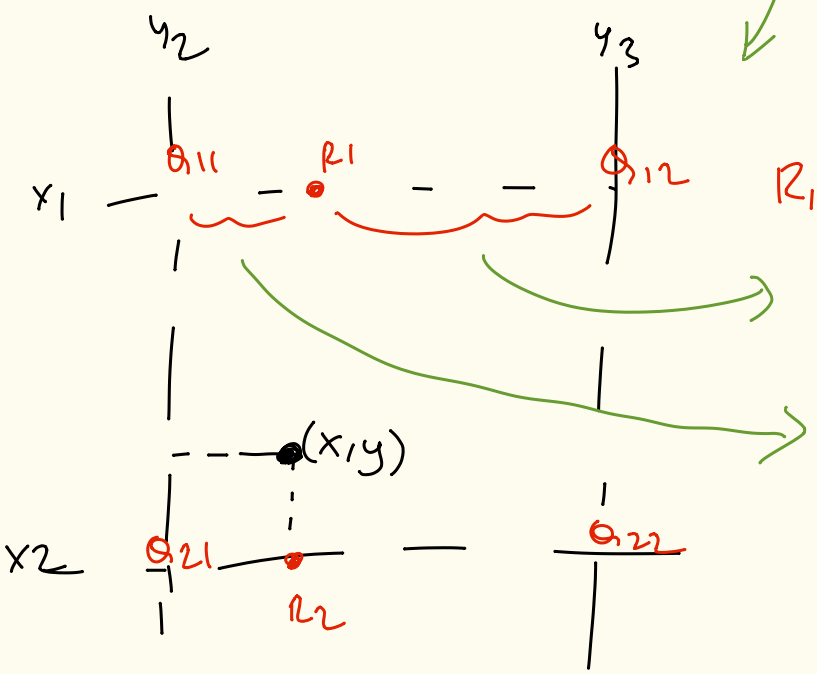
en yakın komşular.

Ex/ Assume that $T^{-1}(2,3) \rightarrow (x,y) = (1.7, 2.3) = p$

↓

∝ Yuvarlayarak interpolation denenebilir.





Bu kısmın mesafeler daha uzun olduğu için daha kısık olan değerle (Q11) çarptık.

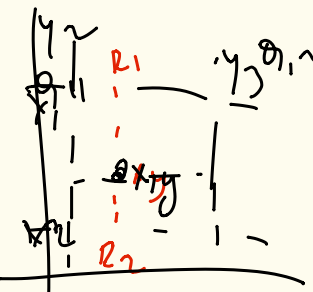
$$R_1 = \frac{y_3 - y}{y_3 - y_2} \cdot Q_{11} + \frac{y - y_2}{y_3 - y_2} \cdot Q_{12}$$

Q → yoğunluk sensörü.

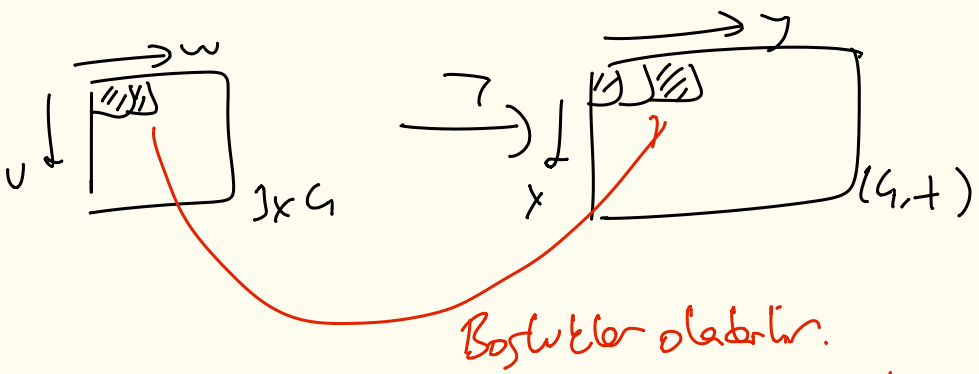
$$R_2 = \frac{y_3 - y}{y_3 - y_2} \cdot Q_{21} + \frac{y - y_2}{y_3 - y_2} \cdot Q_{22}$$

$$P = \frac{(x_2 - x)}{x_2 - x_1} \cdot R_1 + \frac{(x - x_1)}{x_2 - x_1} \cdot R_2$$

$$R_1 = \frac{y_3 - y}{y_3 - y_2} \cdot Q_{11} + \frac{y - y_2}{y_3 - y_2} \cdot Q_{12} \qquad R_2 \cdot -$$



$$P = R_1 \cdot \frac{x - x_2}{x_1 - x_2} + R_2 \cdot \frac{x_1 - x}{x_1 - x_2}$$



outputlar input

yapsak boşluk

sorun olmazdı

Boşluklar olabilir.

Interpolation bu yüzden yapıyoruz

Linear mapping

$$g[x,y] = f[x,y] + b$$

b: scalar value.

} Changing brightness.

$$g[x,y] = a \cdot f[x,y]$$

} Changing contrast.

$a > 1$ increase contrast

$a < 1$ decrease ''

20	150
60	200

$+ 15 =$

35	165
75	215

brightness addition

$a = 1.5$

30	225
90	255

(+15, +75)

contrast addition

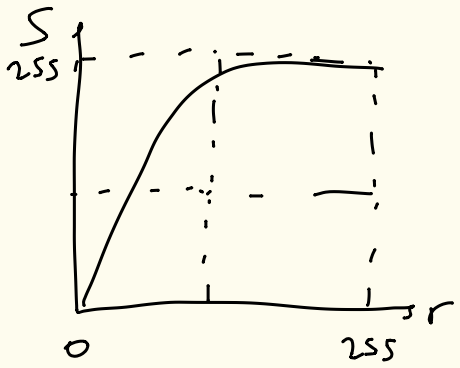
Negative transformation:

bright $\leftrightarrow$ dark

$$S = L - 1 - r$$

- S: output value.
- L: max value 255
- r: original value.

Non-linear mapping:

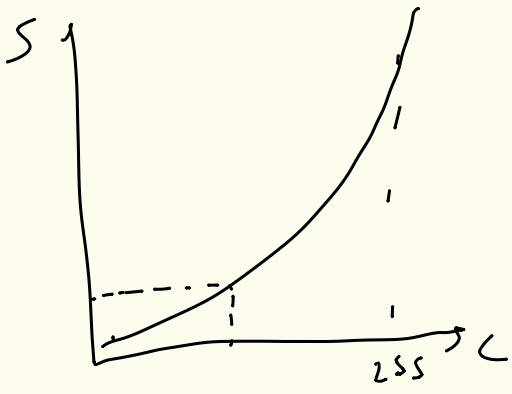


Logarithmic function:

$$S = c \cdot \log(1 + r)$$

- c: coefficient
- r: input

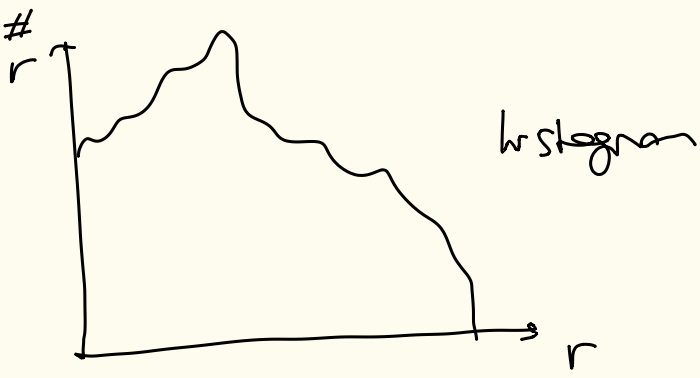
Dark becomes whiter } Dark areas appear lighter while light areas change less.



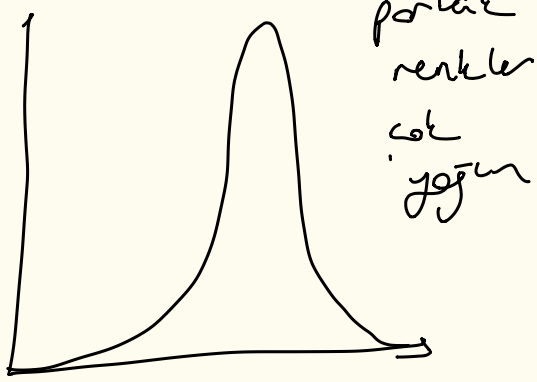
Exponential function:

$$S = c \cdot (e^r - 1)$$

White becomes darker. } As the intensity grows, the output increases rapidly, resulting stronger effect in bright regions, and minimal change in darker regions.



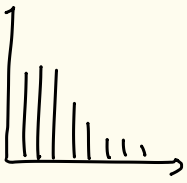
histogram



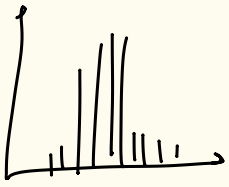
portal
renkler
cok
yogun

Histogram processing

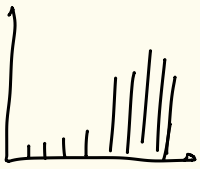
Dark images



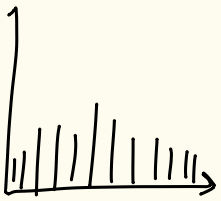
Low contrast image



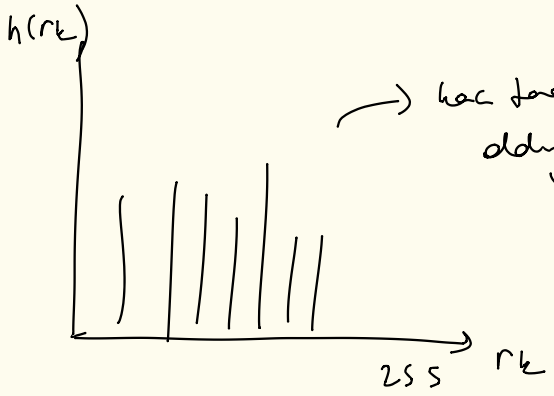
Bright images

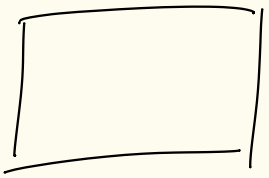


High contrast ..



$h(r_k) = n_k \rightarrow$ # of pixels at r_k value.

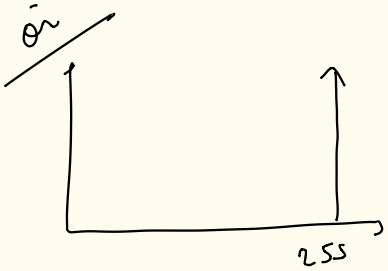




$$p(r_k) = \frac{h(r_k)}{M \times N}$$

↓

Prob. density function.



full beyaz fotoğraf.

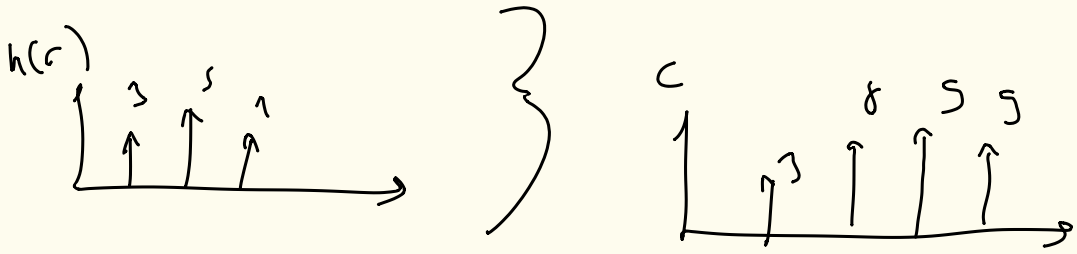
$$c_i = \sum_{i=0}^{j=3} h_i$$

$c_0 = h_0$

$c_1 = c_0 + h_1 = h_1 + h_0$

$c_2 = h_1 + h_2 + h_8 = c_1 + h_2$

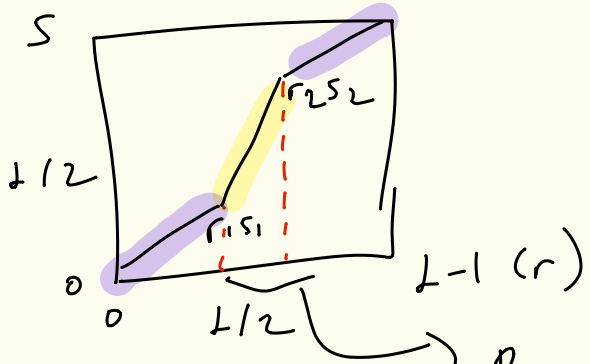
Cumulative Histogram



$$p(j) = \frac{c_j}{M \cdot N}$$

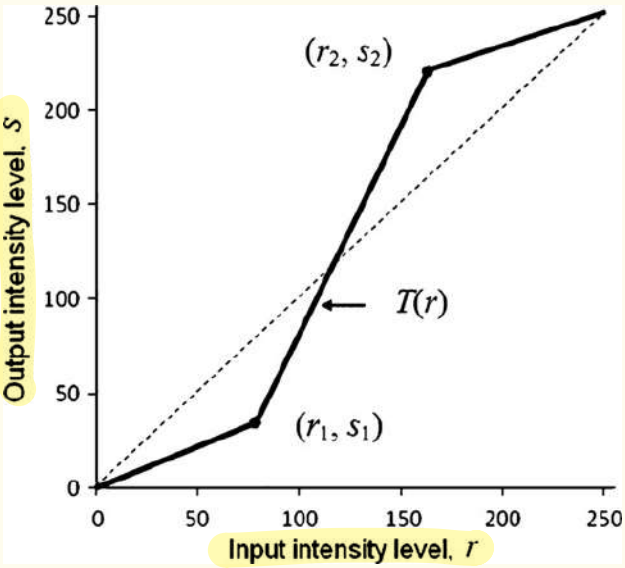
Cumulative Density function

Contrast stretching (Histogram Stretching)



Bu anlatılar contrast artırdım

Histogram yayılınca daha iyi görüntü → High contrast.



r_1, r_2 → Input intensity levels

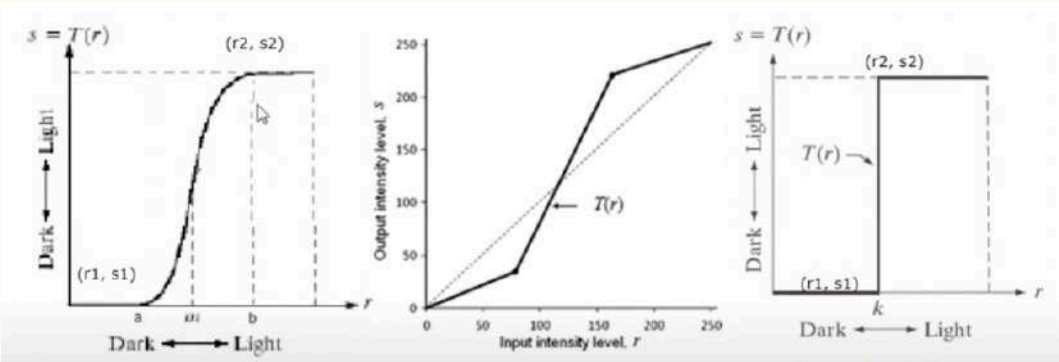
s_1, s_2 → Output " "

1-) Low intensity stretching (0 to r_1); Making dark regions even darker.

2-) Middle " " (r_1 to r_2); Mapping input values to wider range of output values.

3-) High " " (r_2 to 255); Making light regions even lighter.

- Improves contrast by stretching intensity values between r_1 and r_2.
- Useful for images with low contrast where details are not well-defined.



Smooth transitions

Sharp transitions
Good for edges

Only 2 colors
Binary images.

$$I_{new} = \frac{(I_{original} - I_{min}) \cdot 255}{I_{max} - I_{min}}$$

	Original Intensity	Stretched Intensity
Min →	50	0
	75	42
	100	85
	125	127
	150	170
	175	212
Max →	200	255

for example for 50:

$$I_{new} = \frac{(50-50) \cdot 255}{200-50} = 0$$

for 125:

$$\frac{(125-50) \cdot 255}{200-50} = 127$$

for 200:

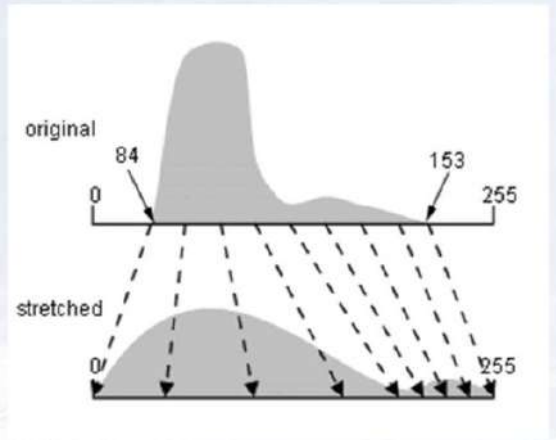
$$\frac{(200-50) \cdot 255}{200-50} = 255$$

Darkest pixels (50) mapped to 0.

Lightest " (200) " " 255.

↓

Now we spanned the full intensity range (0-255), increased contrast.



Histogram Equalization

1-) Create cumulative histogram

2-) Map the histogram to full intensity range.

→ Calculate Cumulative Distribution function

$$P(i) = \frac{\text{count of intensity } i}{\text{total num of pixels.}}$$

Intensity	Count	Probability
0	4	$\frac{4}{16} = 0.25$
1	2	$\frac{2}{16} = 0.125$
2	3	$\frac{3}{16} = 0.1875$
3	2	$\frac{2}{16} = 0.125$
4	1	$\frac{1}{16} = 0.0625$
5	2	$\frac{2}{16} = 0.125$
6	1	$\frac{1}{16} = 0.0625$
7	1	$\frac{1}{16} = 0.0625$

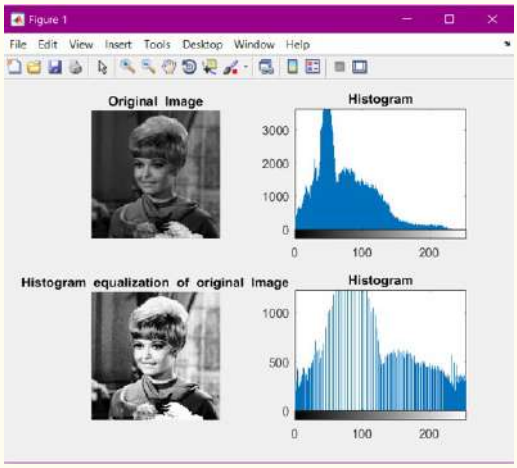
Intensity	Probability	Cumulative Probability (CDF)
0	0.25	0.25
1	0.125	0.375
2	0.1875	0.5625
3	0.125	0.6875
4	0.0625	0.75
5	0.125	0.875
6	0.0625	0.9375
7	0.0625	1.0

→ Map the CDF to new intensity levels } CDF × max intensity

$$\text{New Intensity} = \text{round}(\text{CDF} \times (\text{max intensity}))$$

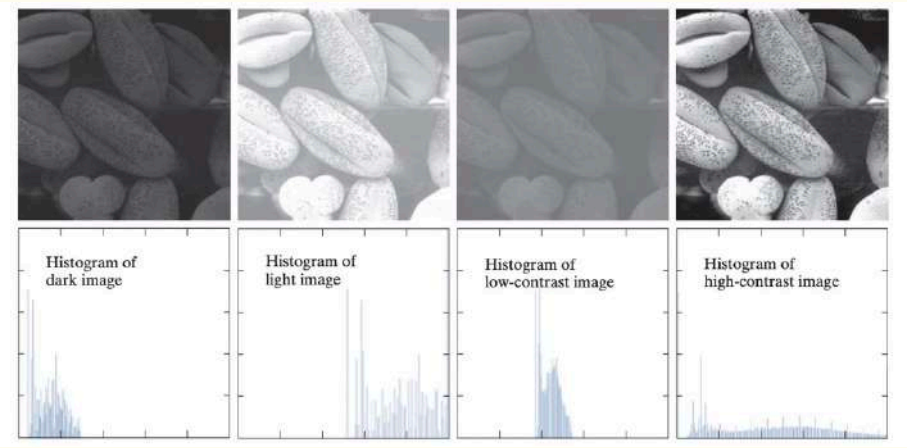
- For intensity 0: New Intensity $= \text{round}(0.25 \times 7) = 2$
- For intensity 1: New Intensity $= \text{round}(0.375 \times 7) = 3$
- For intensity 2: New Intensity $= \text{round}(0.5625 \times 7) = 4$
- For intensity 3: New Intensity $= \text{round}(0.6875 \times 7) = 5$
- For intensity 4: New Intensity $= \text{round}(0.75 \times 7) = 5$
- For intensity 5: New Intensity $= \text{round}(0.875 \times 7) = 6$
- For intensity 6: New Intensity $= \text{round}(0.9375 \times 7) = 7$
- For intensity 7: New Intensity $= \text{round}(1.0 \times 7) = 7$

3-) HE redistributes intensities more evenly across the range.



↳ Dark regions became lighter,

high regions became darker.



Another example:

```
10  20  20  30
30  30  40  50
50  60  70  80
80  90  90  100
```

Intensity	Frequency	Cumulative Frequency	CDF
10	1	1	1/16 = 0.0625
20	2	3	3/16 = 0.1875
30	3	6	6/16 = 0.3750
40	1	7	7/16 = 0.4375
50	1	8	8/16 = 0.5000
60	1	9	9/16 = 0.5625
70	1	10	10/16 = 0.6250
80	1	11	11/16 = 0.6875
90	2	13	13/16 = 0.8125
100	1	14	14/16 = 0.8750

Using the CDF, we apply the histogram equalization formula:

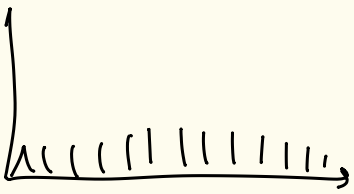
$$T(I) = \text{round}\left((L-1) \cdot CDF(I)\right)$$

Assuming $L = 256$ (for an 8-bit image), the transformation will be:

- For intensity 10: $T(10) = \text{round}(255 \cdot 0.0625) = 16$
- For intensity 20: $T(20) = \text{round}(255 \cdot 0.1875) = 48$
- For intensity 30: $T(30) = \text{round}(255 \cdot 0.3750) = 96$
- For intensity 40: $T(40) = \text{round}(255 \cdot 0.4375) = 111$
- For intensity 50: $T(50) = \text{round}(255 \cdot 0.5000) = 128$
- For intensity 60: $T(60) = \text{round}(255 \cdot 0.5625) = 144$
- For intensity 70: $T(70) = \text{round}(255 \cdot 0.6250) = 160$
- For intensity 80: $T(80) = \text{round}(255 \cdot 0.6875) = 176$
- For intensity 90: $T(90) = \text{round}(255 \cdot 0.8125) = 208$
- For intensity 100: $T(100) = \text{round}(255 \cdot 0.8750) = 224$

Non-linear mapping → Purpose is flatten the histogram

→ HE redistributes the intensity levels to get a flatter histogram.



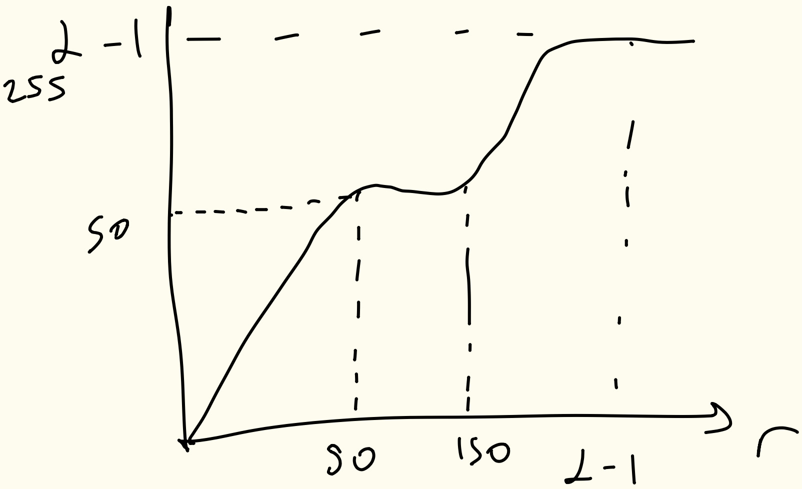
→ HE increase the contrast for the most frequently occuring intensity level.

Reduce the contrast in the less popular part of the image.

r → intensity in an image

s → output intensity.

$s = T(r)$ = output intensity.



Let $P_r(r)$ and $P_s(s)$ denotes the pdf at r and s

$$P_r(r_k) = \frac{n}{m \cdot n}$$

$$s_k = T(r_k) = \frac{L-1}{mn} \sum_{j=0}^{k} n_j$$

Cumulative density dır.

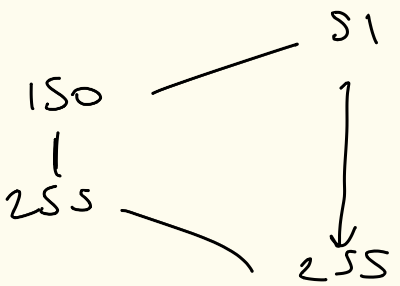
100-120 arası pixel değerimiz yok.

↳ olmayan pixelleri sadece 1 pixel ile ifade ediyorum.

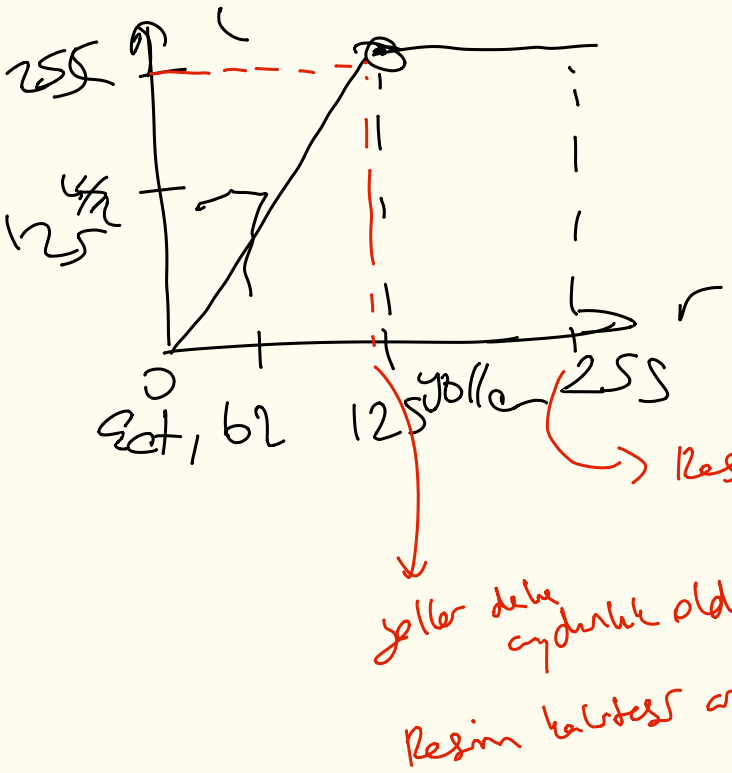
∝ Cumulative ile mapping yaptık.

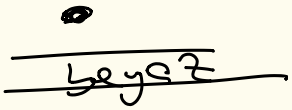
150 → 51

255 → 255



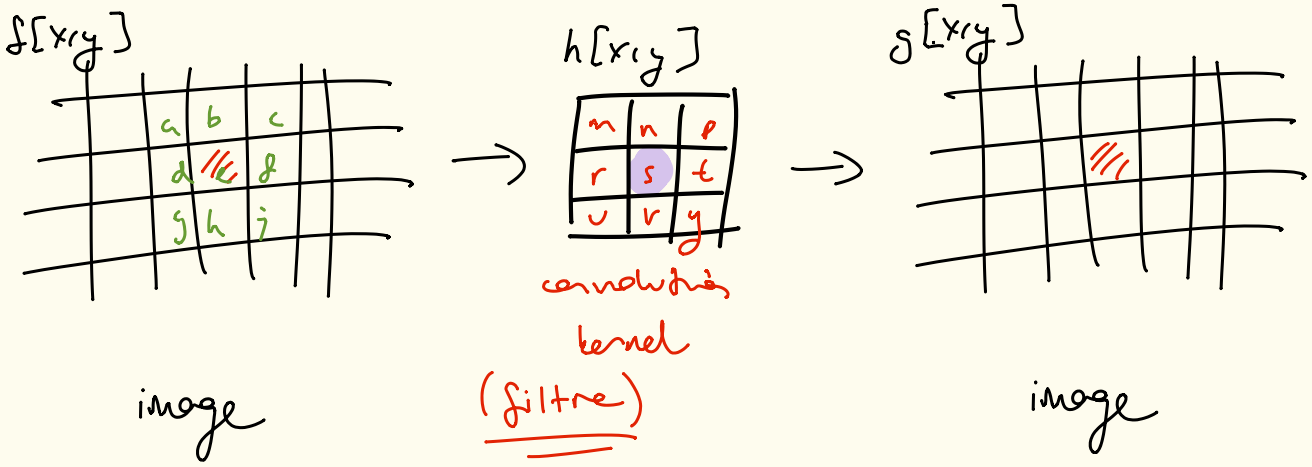
contrastı yaydım.





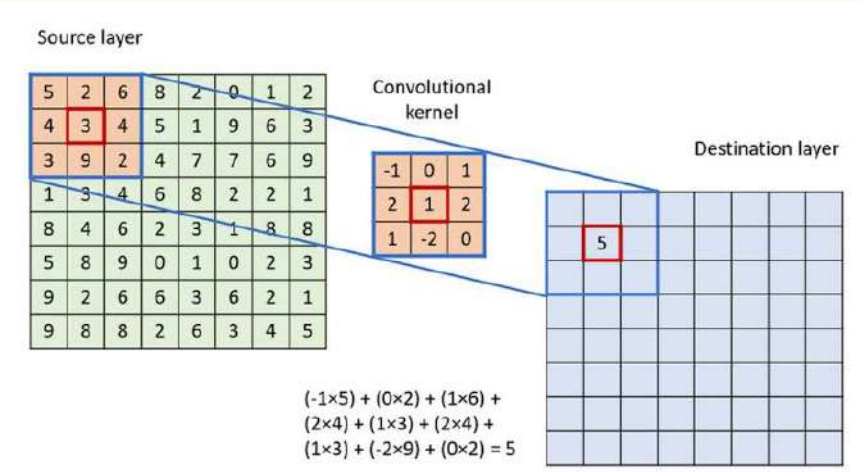
Spatial Filtering

* We used filters / kernels / mask / templates to filter an image.



$$\sum_{s=-a}^{a}\sum_{t=-b}^{b} w[s,t]\cdot f[x+s, y+t]$$ convolution

$$= a\cdot m + b\cdot n + c\cdot p + \cdots + j\cdot y$$



1-) Linear filters:

Low-pass } Smoothing and noise reduction → Mean, Gaussian

High-pass } Edge detection and sharpening → Sobel, Laplacian

* Low pass filtering toplamı 1 olur =

Mean filter:

$\frac{1}{9}$ [1 1 1 / 1 1 1 / 1 1 1] → 9 piksel toplayıp 9'a bölüyorsun → çok beyaz olur!!

O yüzden 1/9 yapıyoruz.

* Low-pass filtering } Reduces noise and detail. Creates blurred effect.

Gaussian filter:

$\frac{1}{16}$ [1 2 1 / 2 4 2 / 1 2 1] → In center, we contribute more.

Sobel filter:

Horizontal edge detection:

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

Vertical " "

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

- The resulting gradients combined to determine edge strength.
- Gradient filter calculates the 1st derivative (gradient) of the image intensity to detect edges. Sobel is specific type of gradient filters.

Laplacian Filter:

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

- Detects edges by calculating 2nd derivative of intensity.

2-) Order Statistics Filters:

Median Filter:

0	100	255
50	255	10
255	0	60

Sort: 0, 0, 10, 50, 60, 100, 255, 255, 255

Median

↳ Replace it with central pixel.

- Good for salt-pepper noise. It preserves the edges because it doesn't make averaging.

Max filter: To highlight bright spots, expand white areas.

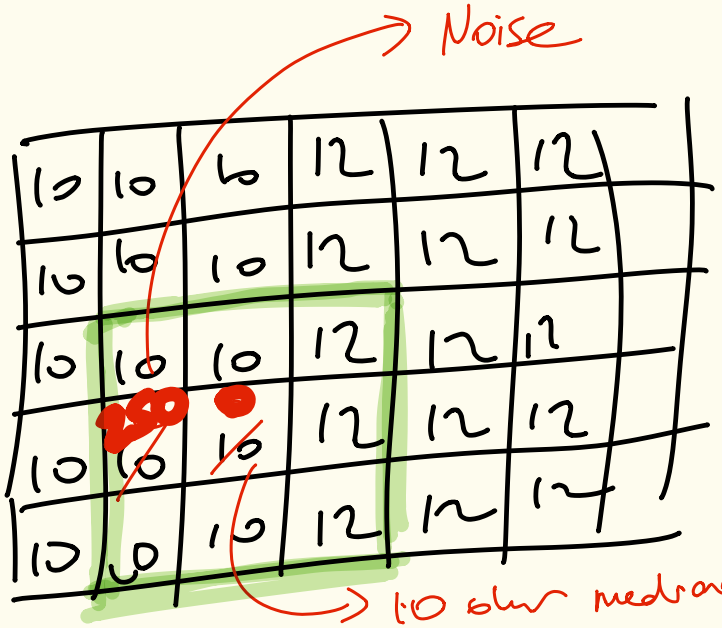
Min filter: " " dark areas.

Rank filters (order statistics)

median
max
min

} You rank the pixels values in kernel from smallest to largest and take the min/max/median.

- Median ones are good at removing "salt and pepper".



Order:

→ 0 10 10 .... 12 12 ... 250

- You want to remove only the noises.
- Low pass filtering (1/9) uygularsan ortası 40 oluyor. (0 dan)
  - ↳ Silindir oluyor. Ama 10 ve 12 yükseliyor.

Sharpening spatial filters

→ to enhance the sharpness in an image we use first and second order filters (derivative)

1st Derivative: Zero in constant areas.
Non-zero at the onset and along the edge.

First derivative of 1D signal:

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

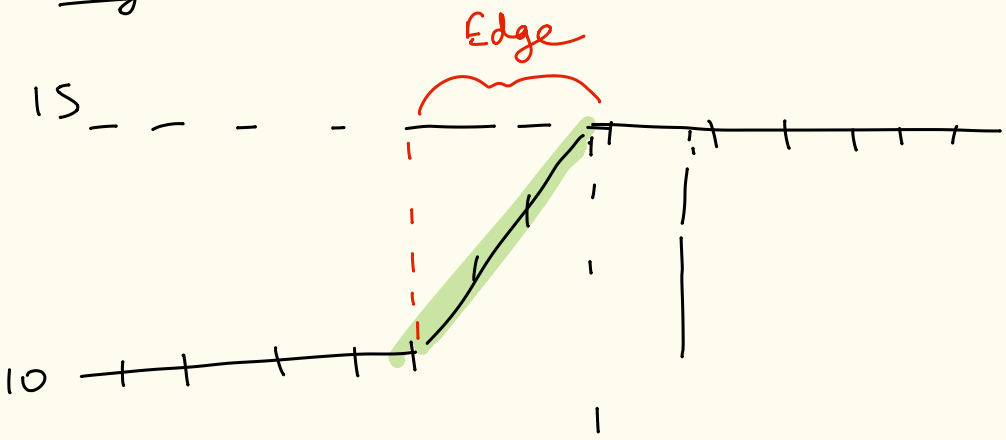
First " " 2D image:

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

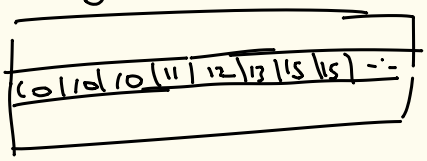
$\hookrightarrow f(x, y+1) - f(x,y)$

$\hookrightarrow f(x+1, y) - f(x,y)$

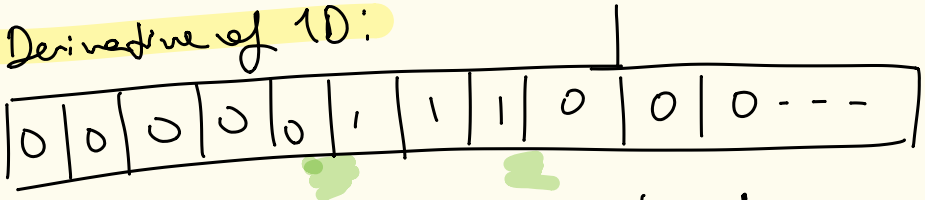
Image line:



1D image:



Derivative of 1D:



$f(x+1) - f(x)$

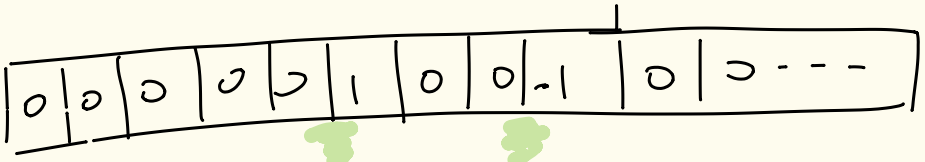
↳ Thick edges.

(If I take derivative of this again (2nd derivative)

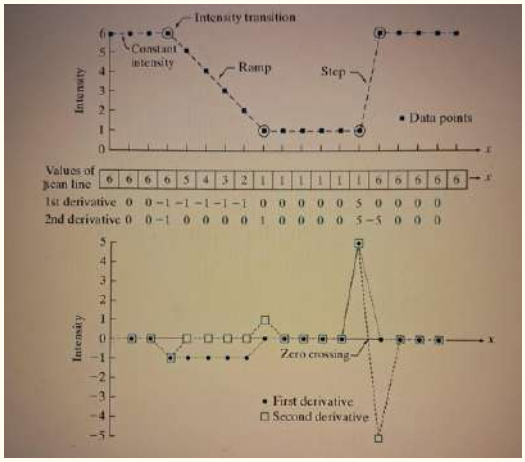
↳ It is zero in constant area and along the ramp.

It is non zero of onset or edges.

2nd Derivative:



↳ Thin edges



1st derivative (Gradient) → Creates thick edges

- Use for highlighting general transition zones.
- It gives a broader location of edges

2nd " (Laplacian) → Thin but double edges.

- More precise locating exact edge positions.

2nd Derivative:

For 1D:

1st → $f'(x) = f(x+1) - f(x)$

2nd → $f''(x) = f(x+1) + f(x-1) - 2f(x)$

For 2D:

2d derivative:

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

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

→ The laplacian is the sum of the 2nd partial derivatives in both directions.

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

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

→ Represent the formula in a kernel:

0	1	0
1	-4	1
0	1	0

Gradient filters :

$$\nabla f = \text{Grad}(f) = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

∝ We can use sobel gradient filters. → High pass filter.

X – Direction Kernel

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

Y – Direction Kernel

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

∝ Detects changes in intensity horizontally

∝ Highlighting vertical edges.

∝ Tan versi

$$\text{Magnitude} = |g_x| + |g_y| \simeq \sqrt{g_x^2 + g_y^2}$$

$$\text{edge direction} = \tan^{-1}\left(\frac{g_y}{g_x}\right) \rightarrow \text{perpendicular to the edge.}$$

Example

$$\begin{bmatrix} 100 & 100 & 100 \\ 150 & 150 & 150 \\ 200 & 200 & 200 \end{bmatrix}$$

Gradient x

$$I_x = (-1 \cdot 100) + (0 \cdot 100) + (1 \cdot 100)$$
$$+ (-2 \cdot 150) + (0 \cdot 150) + (2 \cdot 150)$$
$$+ (-1 \cdot 200) + (0 \cdot 200) + (1 \cdot 200) = 0$$

} No vertical edge.

Gradient y

$$I_y = (-1 \cdot 100) + (-2 \cdot 100) + (-1 \cdot 100)$$
$$+ (0 \cdot 150) + (0 \cdot 150) + (0 \cdot 150)$$
$$+ (1 \cdot 200) + (2 \cdot 200) + (1 \cdot 200) = 400$$

$$\text{Gradient magnitude} = \sqrt{0^2 + 400^2} = 400$$

$$\text{Edge direction} = \tan^{-1}\left(\frac{400}{0}\right) = 90 \text{ degrees}$$

} Edge is horizontal (top to bottom gradient)

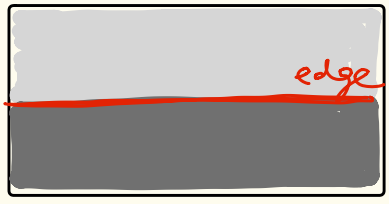


Image sharpening using laplacian:

$\nabla^2 f(x,y)$ → thin edges

$$g(x,y) = f(x,y) + c \cdot \nabla^2 f(x,y)$$

- $g(x,y)$ → sharpened image
- $f(x,y)$ → original image
- c → intensity of sharpening

→ Add the laplacian result back to the orginal image.

Image sharpening using gradient (Unsharp Masking and Highboost Filtering)

1. Blur the original image.
2. Subtract the blurred image from the original (the resulting difference is called the *mask*.)
3. Add the mask to the original.

$f(x,y)$ → orginal image

$\bar{f}(x,y)$ → Blurred image

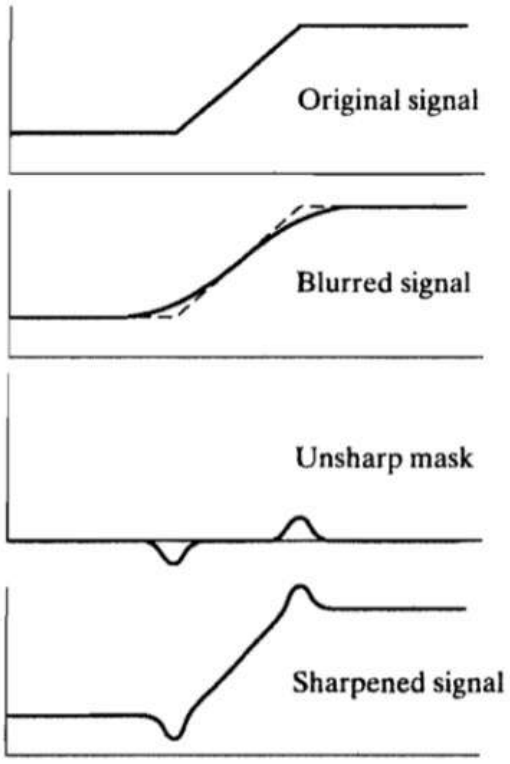
$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

} Mask contains high-frequency components (edges)

$$g(x,y) = f(x,y) + k \cdot g_{mask}(x,y)$$

$k > 1$ high boost filtering } Edges and details boost more strong, higher sharpening

$k = 1$ unsharp image } Enhancing edges without overwhelming the image



a b c
d e

FIGURE 3.49 (a) Original image of size 600 × 259 pixels. (b) Image blurred using a 31 × 31 Gaussian lowpass filter with $\sigma = 5$. (c) Mask. (d) Result of unsharp masking using Eq. (3-56) with $k = 1$. (e) Result of highboost filtering with $k = 4.5$.

Gradient based sharpening

$$g(x,y) = f(x,y) + \beta M(x,y)$$

VİZE

1) Grayscale görsel ile edge detection yapılmış halinin görseli yan yana veriliyor. Aşağıdaki filtrelerden hangileri uygulanmıştır veya uygulanmamıştır, hepsini tek tek yorumla. (20p)
   a) Laplacian Kernel
   b) Blur uygulayıp orijinalden çıkarma (Highboost filter)
   c) Önce median filter uygulayıp sonra edge detection uygula
   d) Önce edge detection uygulayıp sonra median filtre uygula

Laplacian → Double and thin edges.

2nd derivative filter for sharpening edges. → "rapid intensity change"

first derivative ile kıyasla daha keskin kenarları ayırt eder. (Sobel)

Highboost → Eğer katsayı yükselirse daha net bir edge detection görebiliriz.

Blur olduğu için laplacian'a göre daha yumuşak edgeler görebiliriz.

Median → edge → Median filter (özellikle salt pepper) noise reduction yöntemidir. Gürültüsüz resimde bu olabilir

Edge → median → Önce edge detection yaparsak, eğer gürültü varsa edgeler gürültü ile karışabilir. Daha sonra ortalama bir görüntü olabilir.

2) Orijinal elma resmi ile blurlenmiş hali yan yana verilmiş. Aşağıdaki filtrelerden hangileri uygulanmıştır veya uygulanmamıştır, hepsini tek tek yorumla. (20p)

a) Mean filter

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

→ Low pass, smoothing filter ✓

Bu olabilir çünkü low frequency çok etkilemezken, high frequency olanları düşürür. Bu yüzden smoothing etkisi olur and we kill the details.

b)

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

→ Laplacian ✗

c)

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

→ Laplacian ✗

High pass

d)

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

→ Sobel ✗

Low pass filter → Smooths the image by removing high-frequency details, resulting in a blurred appearance.

High pass filter → Sharpens the image by increasing high-frequency details, edge ve detayları vurgular.

3) 5x5 matrix veriliyor. (2, 2) konumuna aşağıdaki filtreleri uygulayıp sonuçları yazın. (30p)
   a) Mean filter
   b) Median filter
   c) Sobel filter

Mean →

1	1	1
1	1	1
1	1	1

Median → Sırala ve ortayı al

Sobel →

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

dik edgeler

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

yatay edgeler

Magnitude of point → $\sqrt{G_x^2 + G_y^2}$

Yön → $\arctan(G_y / G_x)$

In a given application, a smoothing kernel is applied to input images to reduce noise, then a Laplacian kernel is applied to enhance fine details. Would the result be the same if the order of these operations is reversed?

∝ Smoothing → Laplacian :

Overall image becomes softer and less detailed.

Then Laplacian can effectively highlight the edges and details that remain.

∝ Laplacian → Smoothing :

Laplacian highlights edges but it also strenghten any noise present in image.

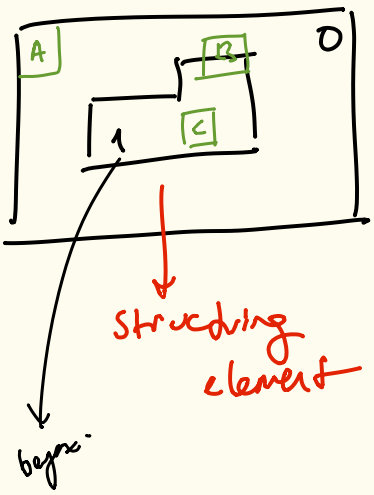
Then smoothing blurs the image, reduces both noises and strong edges created by Laplacian.

14 KASIM

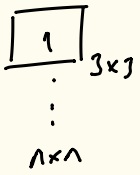
Morphological Image Processing (MIP)

MIP is a non-linear i.p. and usually used on binary images.

- use structring element for that



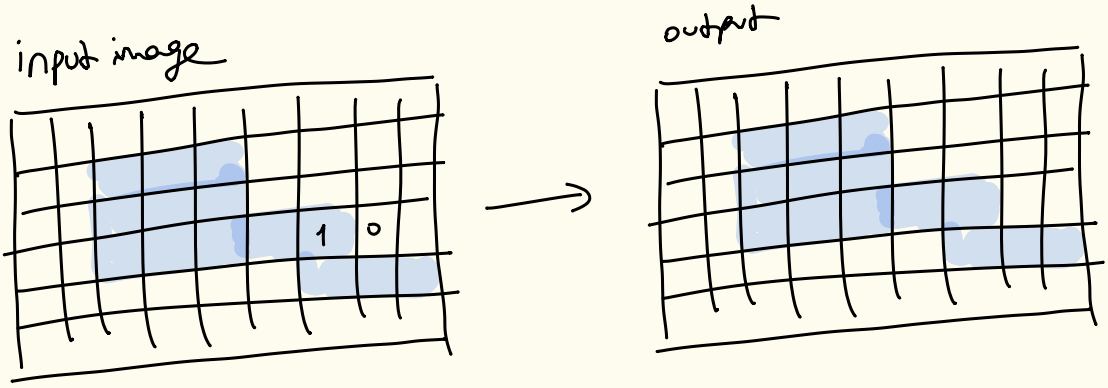
background is 0.



A -> miss the object

B -> hit " "

C -> fit " "



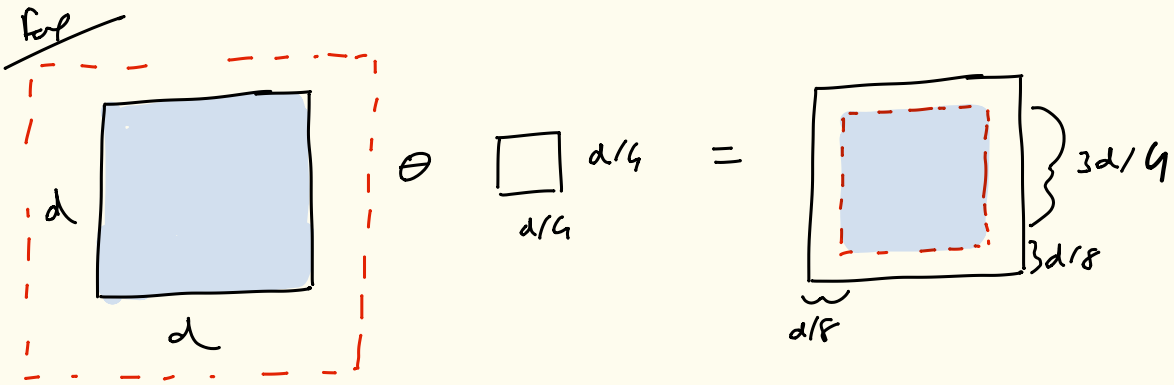
kernel:

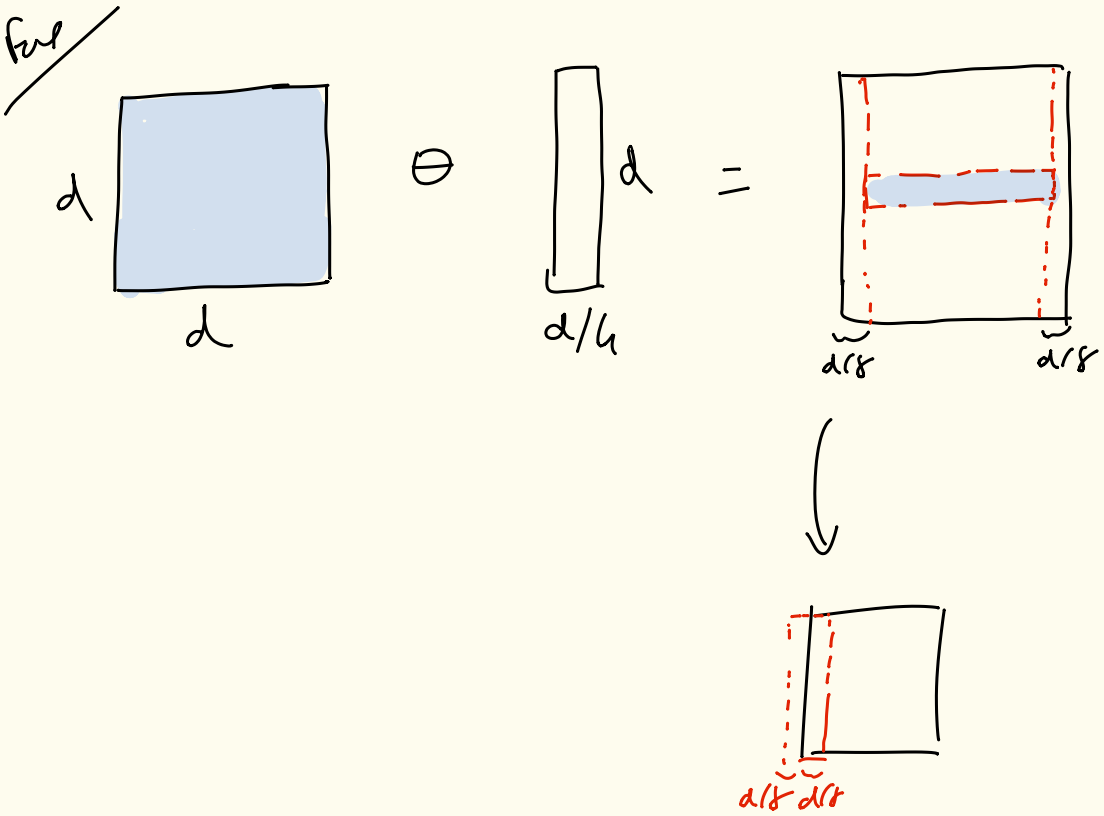
→ Make convolution with passing through in image.

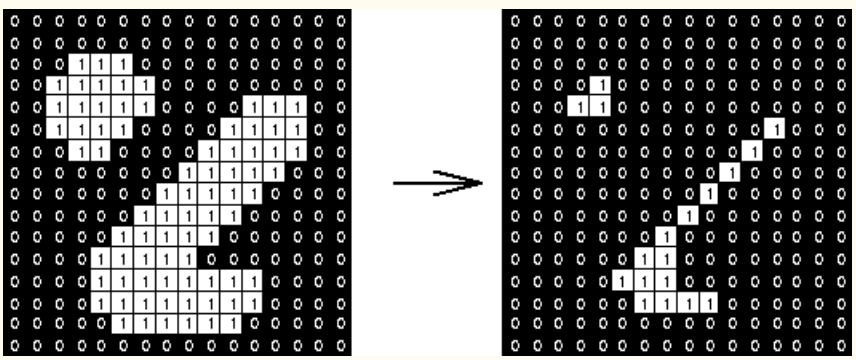
Erosion: with A and B as Z^2, the erosion of A by B, which is defined as

$$A \ominus B = \{ z \mid B_z \subseteq A \}$$

AND operation


for
d
d
⊖
d/4
d/4
=
3d/4
3d/8
d/8

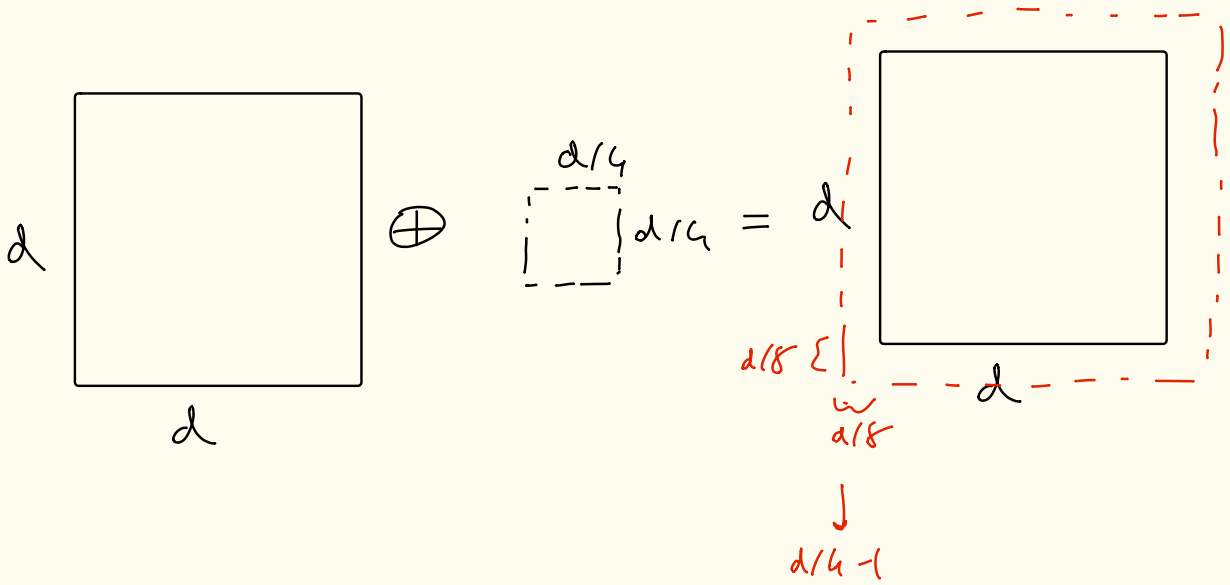

for
d
d
⊖
d
d/4
d/8
d/8
d/8
d/8

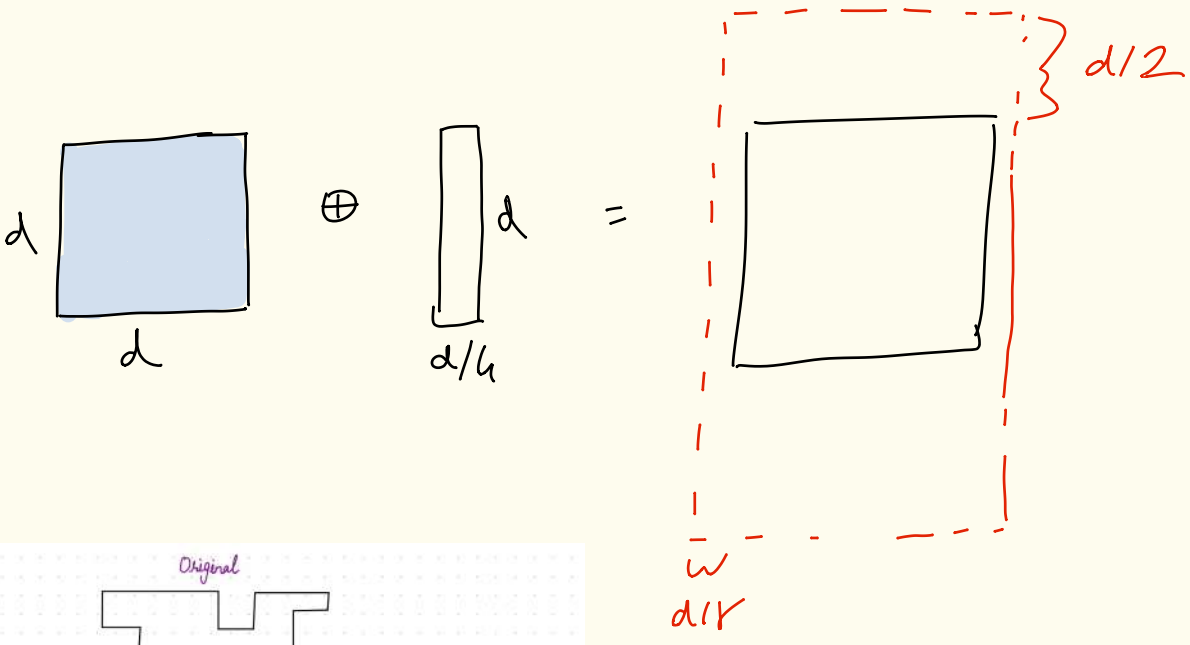


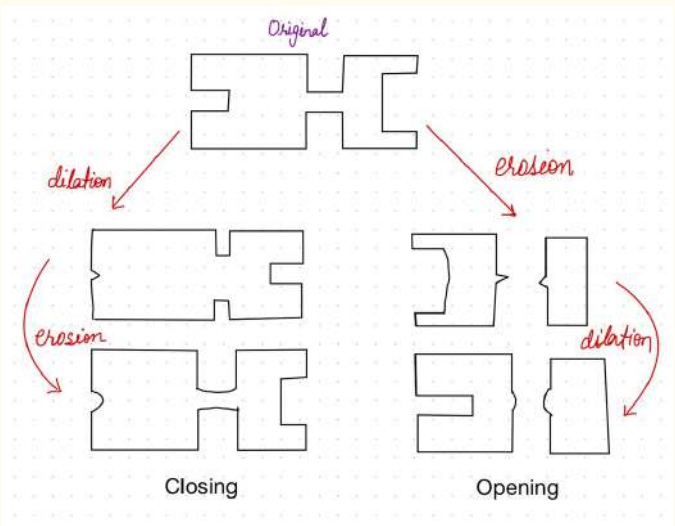
Dilation:

$$A \oplus B \rightarrow \{ z \mid B_z \cap A \neq \phi \}$$

or







Segmentation

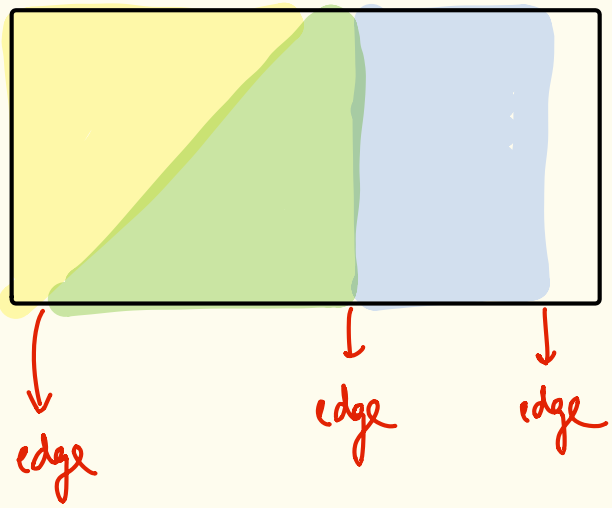
Process of grouping pixels having similar attributes.

↳ Segmentation algorithms divided into 2 groups:

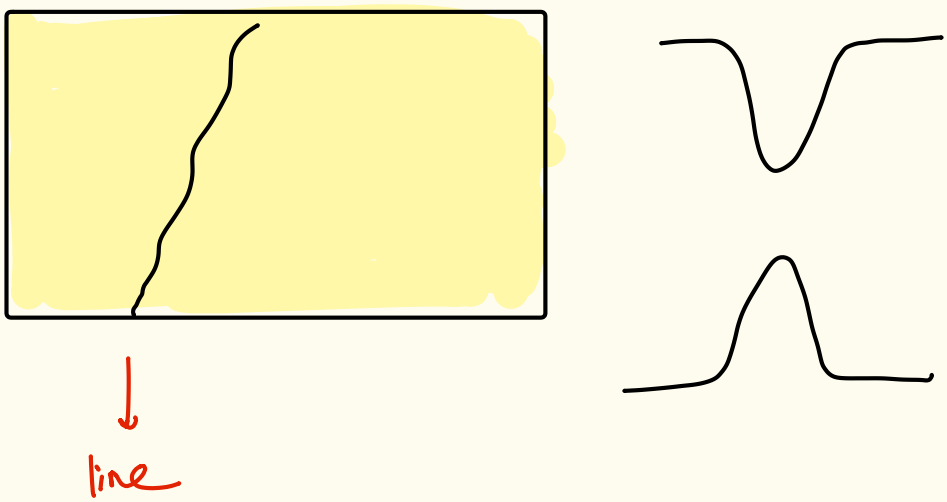
- Contextual
- Non-contextual

Some definitions:

↳ Edge: Pixels at which intensity changes abruptly



↳ Line: It is an edge which intensity of background on either side is higher or lower than edge.



↳ Isolated points: Line whose length and width is equal to $\underline{1}$.

(Point that intensity is different than others)

→ Gradient - 1st
→ Laplacian - 2nd
} for detecting edges

$$\nabla f = \begin{matrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{matrix} \Bigg\} \begin{matrix} g_x \\ \\ g_y \end{matrix} \Bigg\} \quad g_x = \frac{\partial f(x,y)}{\partial x}$$

$$M = \sqrt{g_x^2 + g_y^2}$$

$$\alpha = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

∇f → produce thicker edges

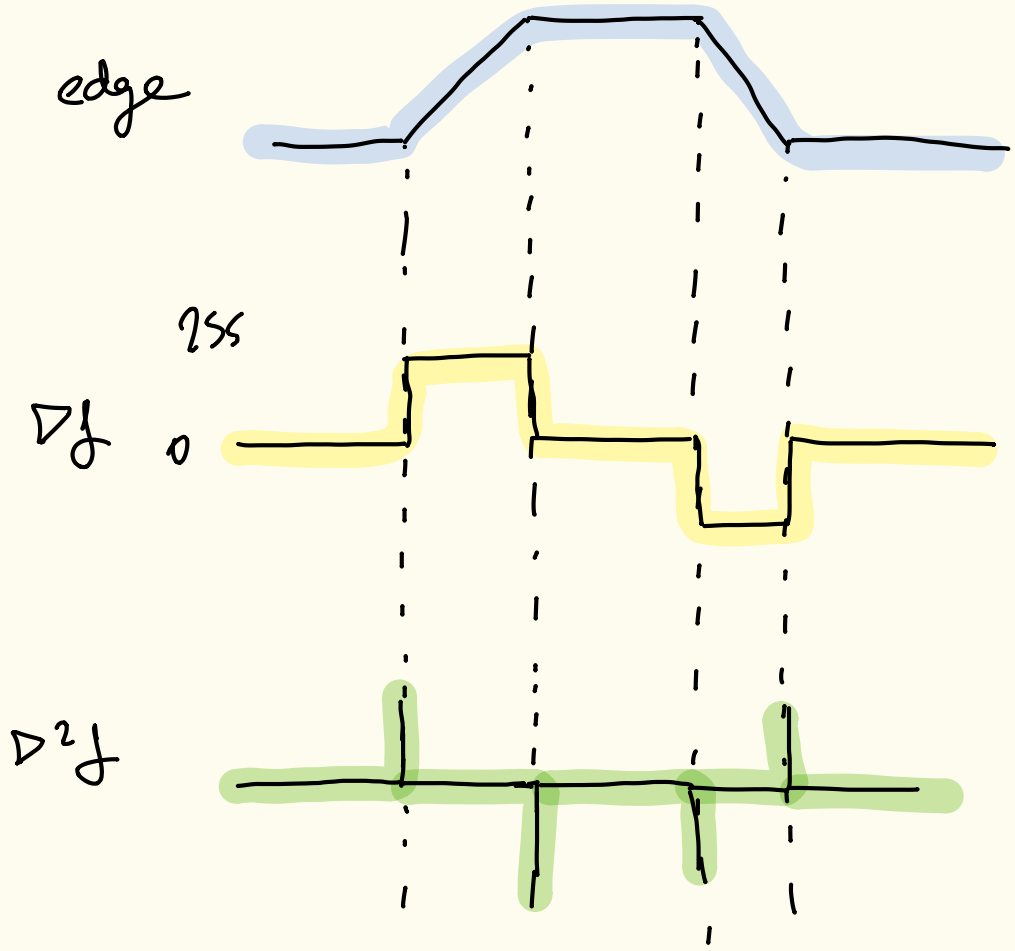
$$\nabla^2 f(x,y) = \frac{\partial f^2}{\partial x^2} + \frac{\partial f^2}{\partial y^2} \Bigg\} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

↳ produce thin lines at both side of edge

↳ respond to finer details

↳ sensitive to noises.

↳ we caged negative values.



$\nabla^2 f \rightarrow$ positive } transition from dark to light ↑ 255 / 0

" $\rightarrow$ negative } " " light to dark ↓ 255 / 0

Detecting Isolated Points:

image: $f(x,y)$

10	11	10	5	10
10	10	200	10	10
11	10	5	10	11
11	11	1	10	11

isolated point

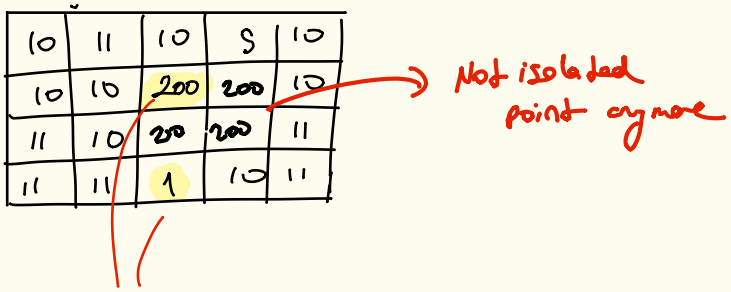
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \approx \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

↳ Increase the effect of isolated point.

$f(x,y) = \nabla^2 f(x,y)$ → apply laplacian

isolated point: $g(x,y) = \begin{cases} 1 & |R(x,y)| \geq T \\ 0 & \text{otherwise} \end{cases}$ ← threshold

1. Apply the Laplacian operator to the image.
2. Calculate the new image $\delta(x, y)$ from the Laplacian result.
3. Use a threshold τ to detect isolated points:
   - Mark pixels as **isolated** if their Laplacian value is greater than or equal to the threshold.
4. If the pixel is no longer significantly different from its neighbors, it is no longer considered isolated.



Line Detection:

$\nabla^2 f$ has stronger response for lines than ∇f.

→ But we don't want double lines. We have to handle them.

1-) Apply Laplacian

2-) Take the absolute value of response. (thick lines)

or

2-) Take the positive one (thin line)

4 Laplacians are isotropic (symmetric in both directions)

↓

We can get directional masks.

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$ —— horizontal lines

or

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$ | vertical lines

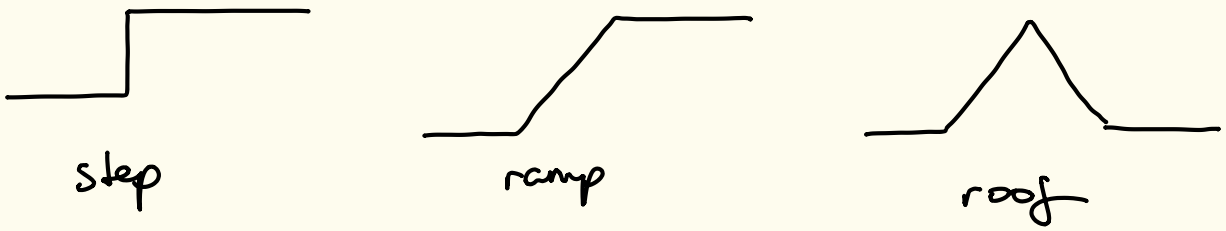
$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$ \ +45°

or

$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$ / -45°

& you may combine them to detect all of the lines.

Type of edges:



→ Edge detection algo. have 3 steps:

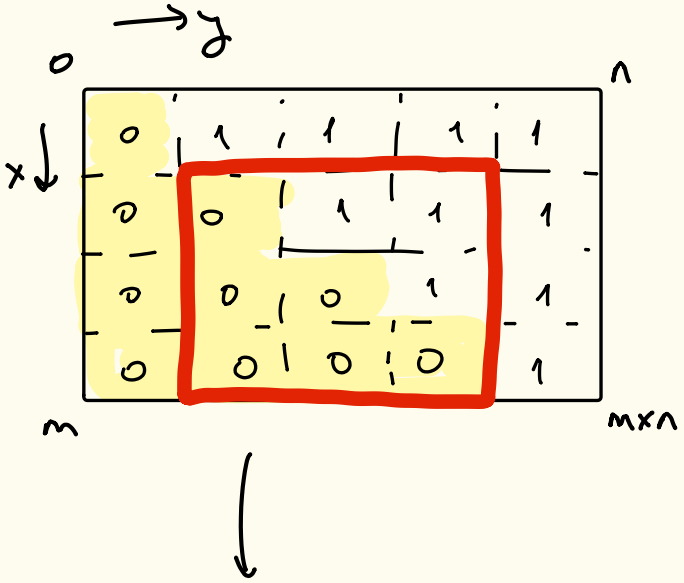
1-) Use low pass filter → image smoothing for noise reduction

2-) Detecting edge points → laplacian, Gradient
Candidate edges

3-) Edge localization → Select from the candidate edge pixels to find true edges

Gradient :

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \qquad M(x,y) = \sqrt{g_x^2 + g_y^2} \qquad \alpha = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$



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

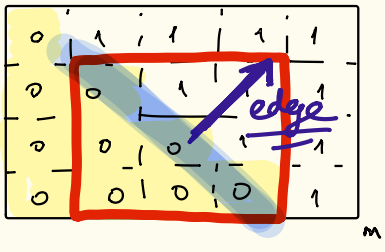
$$g_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

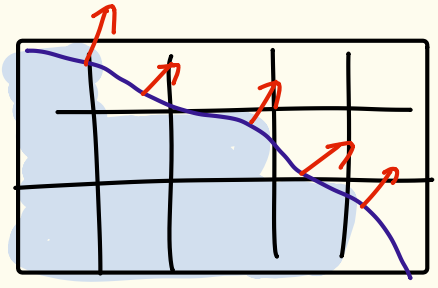
$g_x = -2$

$g_y = 2$

$M(x,y) = 2\sqrt{2}$

$\alpha = \tan^{-1}(-1) = -45°$





suive

↳ You can use thresholding to detect candidate edges.

↳ An edge detection operator should have:

→ Capability of estimating the 1st and 2nd derivative.

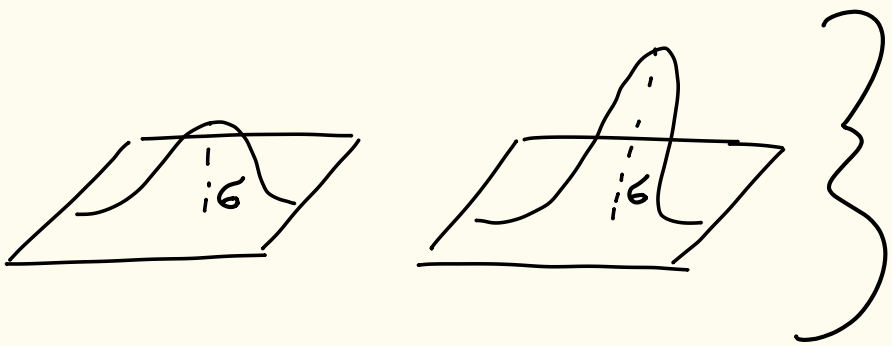
→ Should be tuned to act at any chosen at any scale, so that operators can be used to detect blurry edges and small operators to detect sharp edge.

↳ ayarlanabilir olmalı.

Laplacian of Gaussian (LoG)

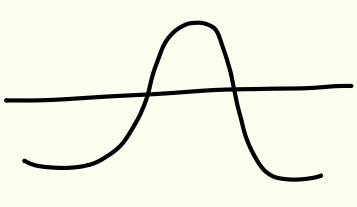
$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)}$$

↳ standart deviation



Low pass filters.

Katsayılar toplamı 1.



High pass

Kenarları belirler diyebiliriz.

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

2nd derivative for both x, y direction

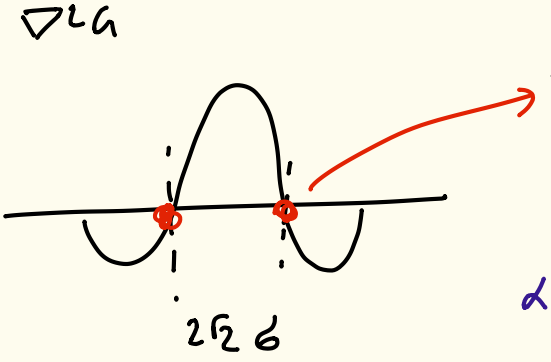
$$\text{LoG Filter} = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] \cdot \underbrace{e^{\frac{-(x^2-y^2)}{\sigma^2}}}_{\text{Gaussian function}}$$

↳ standart deviation controls amount of smoothing

↳ Includes both removing noises and detecting edges → Combined filter.

1-) Gaussian smoothing (reduce noise)

2-) Laplacian (detect edges)



- Zero crossing of LoG occur at

$$x^2 + y^2 = 2\sigma^2$$

↳ Laplacian finds edges by finding zero crossings.

Exp

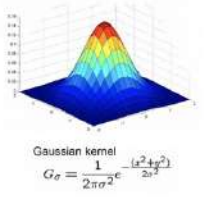
for 5×5 LoG filter:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 1 & -2 & 16 & -2 & 1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

0	−1	0
−1	4	−1
0	−1	0

Laplacian Filter



Gaussian Filter

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

LoG Filter

Marr-Hildreth Edge operator;

Consists of convolving the LoG filter with an image.
and get zero crossings as edges.

$$g(x,y) = \left[\nabla^2 G(x,y) * f(x,y)\right]$$

(* → convolution)

∇^2 and $*$ are linear operators.

$$g(x,y) = \nabla^2\left[G(x,y) * f(x,y)\right] = \nabla^2 G(x,y) * f(x,y)$$

In summary:

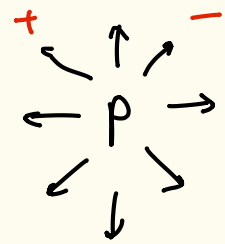
1- Filter the image with Gaussian kernel.

2- Apply Laplacian to smoothed image

3- find zero-crossings in the Laplacian image to detect edges.

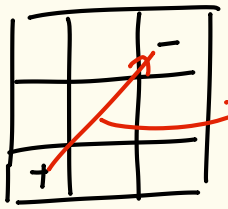
↳ Identifies edges with high accuracy.

Zero-crossing: of pixels implies that sign of at least 2 of its opposite neighbour pixels must be different.



∝ For opposite neighbouring cases.

- left / right
- up / down
- 2 diagonals.

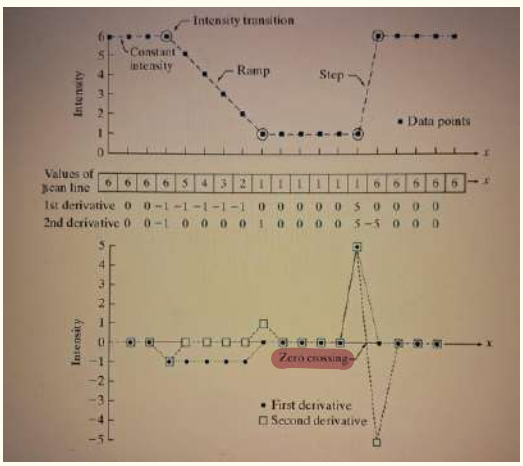


Older geeks.

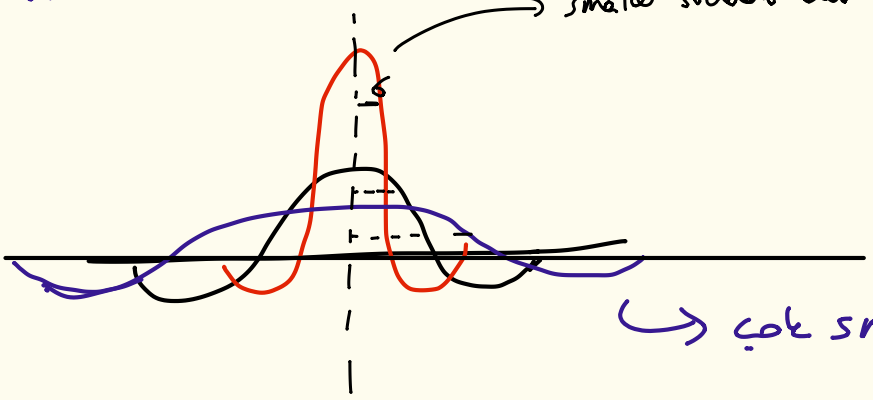
∝ 2nd derivative change sign.

↓

This will be edge if we find zero crossing.



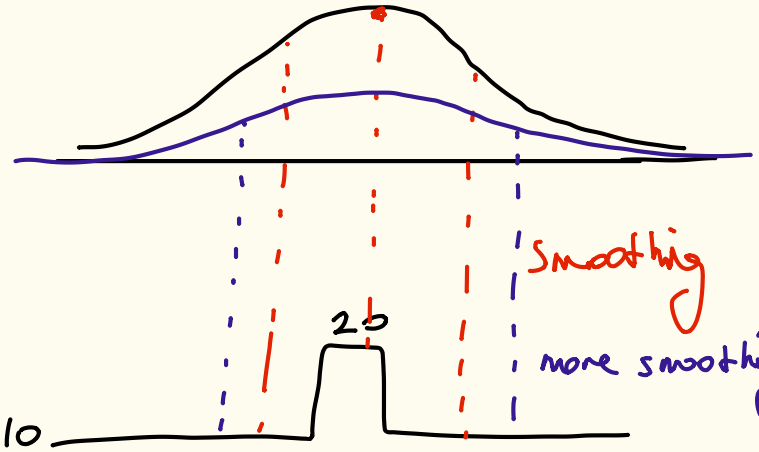
Gaussian curves:



→ smaller standart deviation ⇒ less smoothing

↳ çok smoothing, büyük kenarlar için

Gaussian:



Smoothing

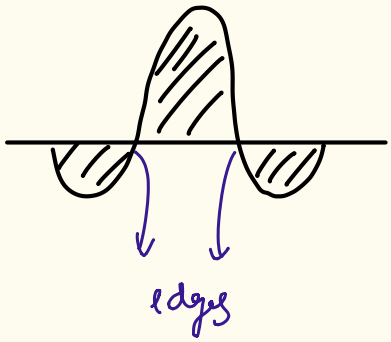
more smoothing with changing standart deviation

↳ Pikseller daha çok yayılır → more smoothing

LoG:

Gaussian + Laplacian

High pass filter new.



∴ LoG → ölçeklenebilir filtre

28 KASIM

Canny Edge Detection:

1-) Smooth image using Gaussian filter → To remove noise + Decrease fine details. (edge somehow)

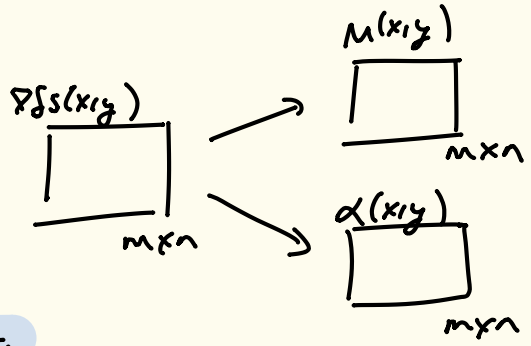
2-) Compute gradient images. → Magnitude
→ Angle

3-) Apply non-maxima suppression to the gradient mag. image

4-) Use double thresholding and connectivity analysis to detect and link the edges.

1-) $f_s(x,y) = h(x,y) * f(x,y)$ } Gaussian

2-) $\nabla f_s(x,y)$ → $M(x,y)$
→ $\alpha(x,y)$



Every pixel has gradient output.

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

→ apply sobel, find g_x and g_y for each pixel.

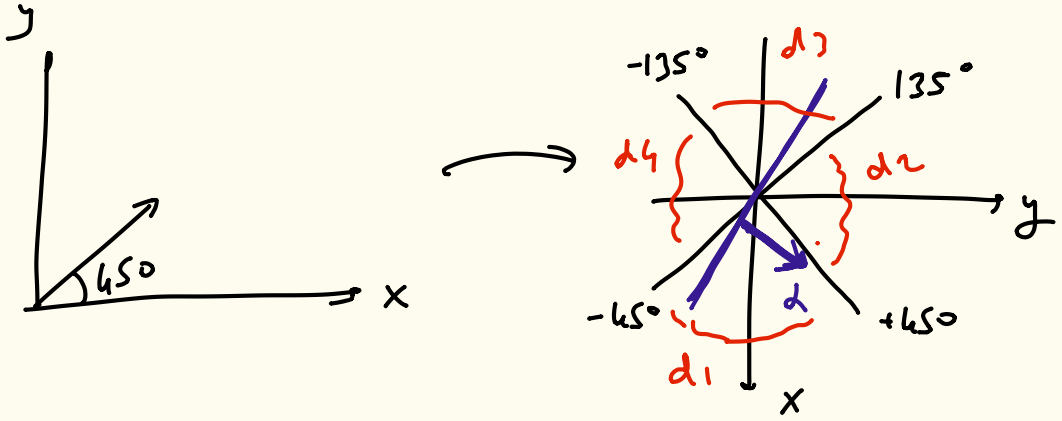
3-) Non-maximum supression → It removes thick edges and remains only the sharpest, thinnes edges.

M(x,y) contains wide ridges (sird) around local maxima

so these ridges should be thinned.

↳ We diwide the areis into basically 4 segments.

(d1, d2, d3, dn)

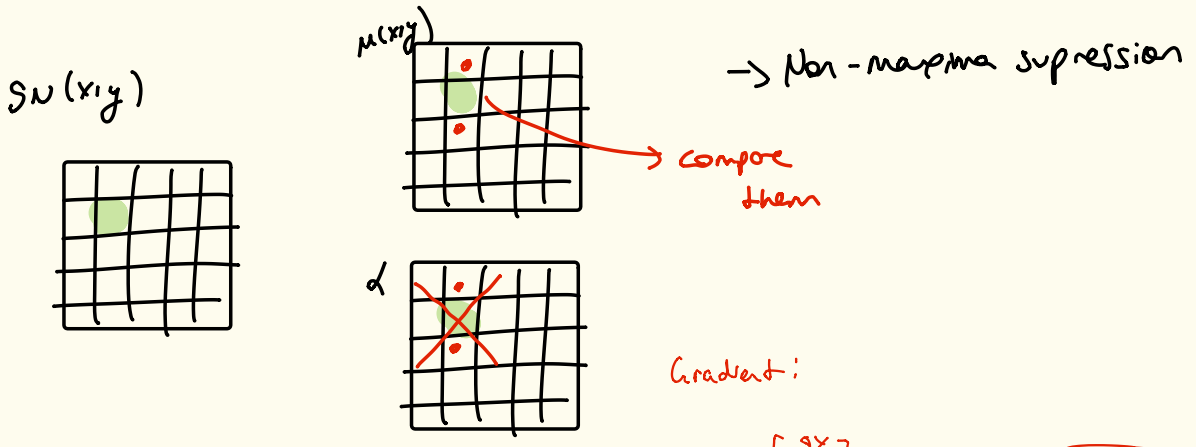


1-) Once you know the "gradient direction α(x,y)" for a pixel, compare the magnitude with its 2 neighbours along the gradient direction.

2-) If the M(x,y) is less than at least one of its 2 neighbours:

$g_N(x,y) = 0$ (suppression), otherwise

$g_N(x,y) = M(x,y)$ (kept the value)



Gradient Magnitude Before NMS (Wide Edges)

$$M(x,y) = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 5 & 8 & 6 & 1 \\ 0 & 3 & 9 & 4 & 0 \\ 2 & 6 & 7 & 5 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

Wide edges.

Gradient Magnitude After NMS (Thin Edges)

$$M_{\text{NMS}}(x,y) = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 5 & 8 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

Edges are thinned by only keeping strongest pixel in the edge direction.

4-)

→ Double threshold to detect true edges

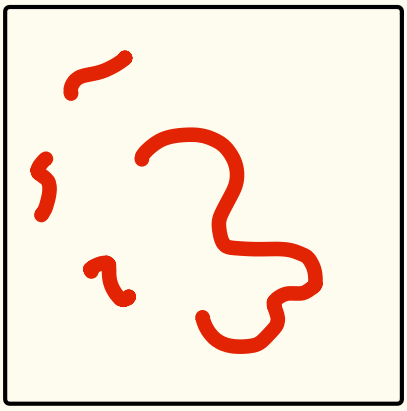
$$g_{NH}(x,y) = g_N(x,y) \geq T_H \quad \text{(strong edges)}$$

$$g_{NHL}(x,y) = g_N(x,y) \geq T_L \quad \text{(strong + weak edges)}$$

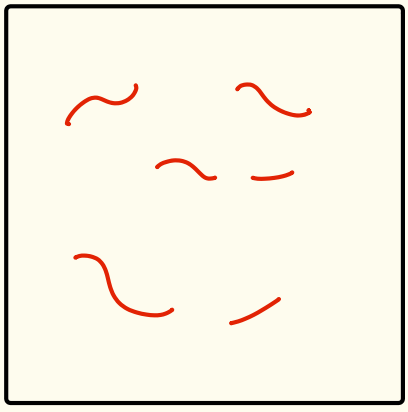
$$g_{NL}(x,y) = g_{NHL} - g_{NH} \quad \text{(weak edges)}$$

∝ Once the strong and weak edges are found, link the weak edges to the strong edges.

Applying edge linking:



g_{NH}



g_{NL}

g_{NH} → true edges

g_{NL} → candidate "

→ Our purpose is complete the gaps → linking

(link the gap in g_{NH})

1-) Locate the next unvisited edge pixel p in $g_{NL}(x,y)$
(weak edge)

2-) If unvisited edge pixel g_{NL} is connected to a strong edge (8-connectivity) pixel, promote it to a strong edge.
(mark)

3-) If all weak edge pixels have visited go step 4,
otherwise go to step 1.

4-) Any weak edges that was not linked a strong edge is discarded,
set to 0.

↳ This ensures that the final edge map contains continuous, connected edges.

Thresholding

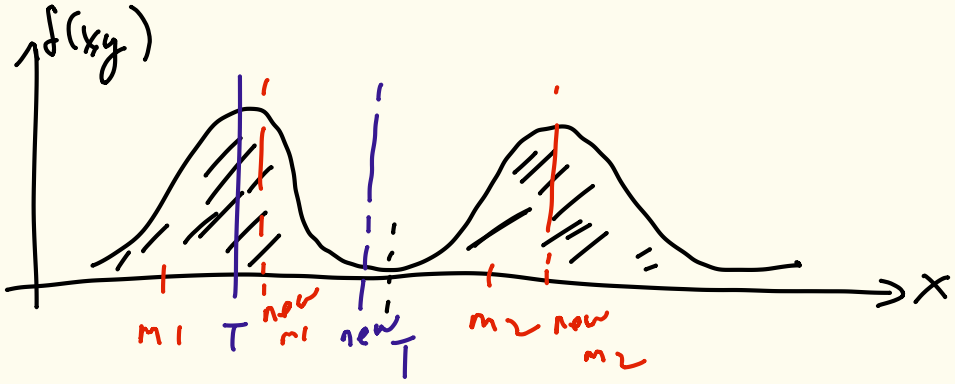
$$g(x,y) = \begin{cases} 1 & f(x,y) > T \\ 0 & " \quad \leq T \end{cases}$$

Multiple T

$$g(x,y) = \begin{cases} a & f(x,y) > T_2 \\ b & T_1 < " \leq T_2 \\ c & " \leq T_1 \end{cases}$$

How to determine threshold T that seperates image into foreground G1 and background G2 based on intensity dist.:

Image Dist.



Iterative method:

1-> Select an initial estimate for T

2-> Compute the group of G1 and G2

$G1 = \{ f(x,y) > T \}$ → set of pixels bigger than T.

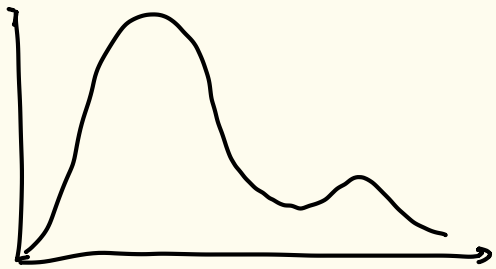
$G2 = \{ f(x,y) \leq T \}$

3-> Compute the mean of G1 and G2 $\begin{cases} m_1 = \text{mean}(G1) \\ m_2 = \text{mean}(G2) \end{cases}$

4-) " the new threshold } $T = \frac{m_1 + m_2}{2}$

5-) Iterate until no improvement on T.

Not suitable for this → we should look variance



↓

Otsu's method

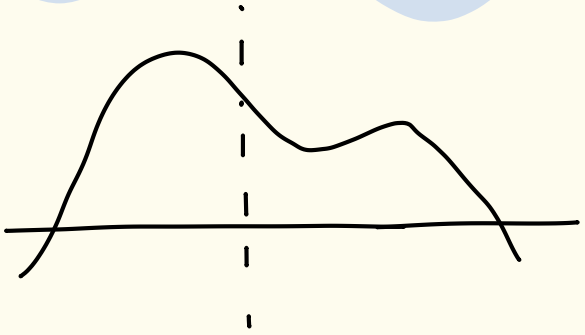
$B \leq G$

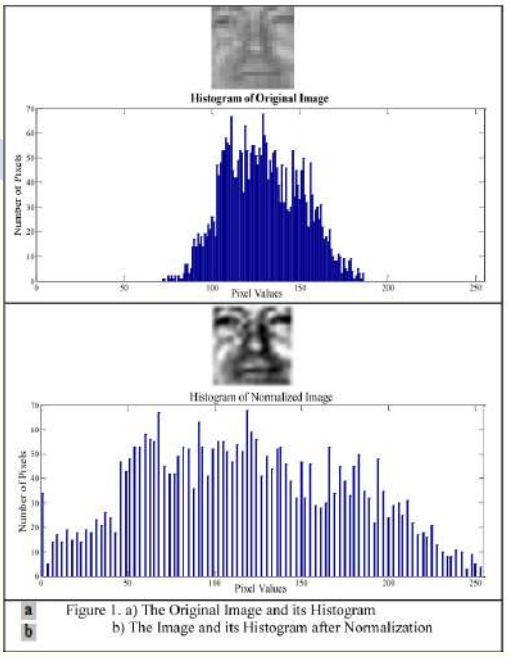
$$H = \cos^{-1}\left(\frac{\frac{1}{2} \cdot a + b}{\sqrt{a^2 + b(G-B)}}\right)$$

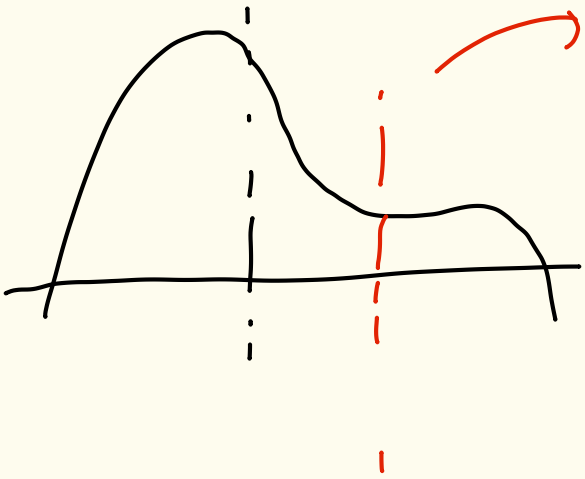
$$S = 1 - \frac{3}{R+G+B} \cdot \min($$

$$I = \frac{R+G+B}{3}$$

V: very



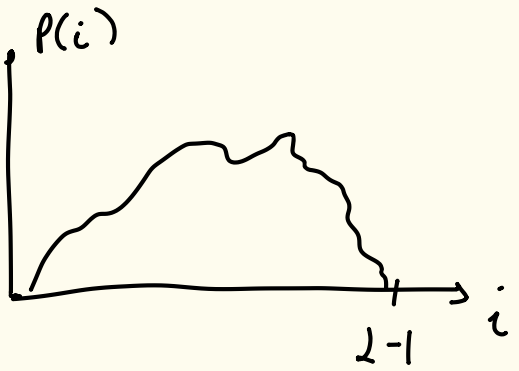




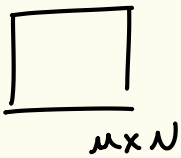
Our purpose is find threshold like this

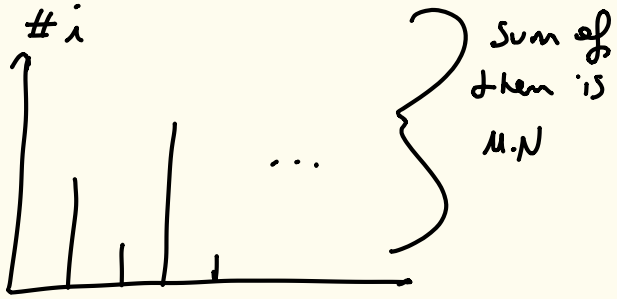
OTSU thresholding

1-) Compute normalized histogram of an image



$$p(i) = \frac{\#i}{M \cdot N}$$





i #i
1 -> one
2 -> 2 one
3 -> 4 one

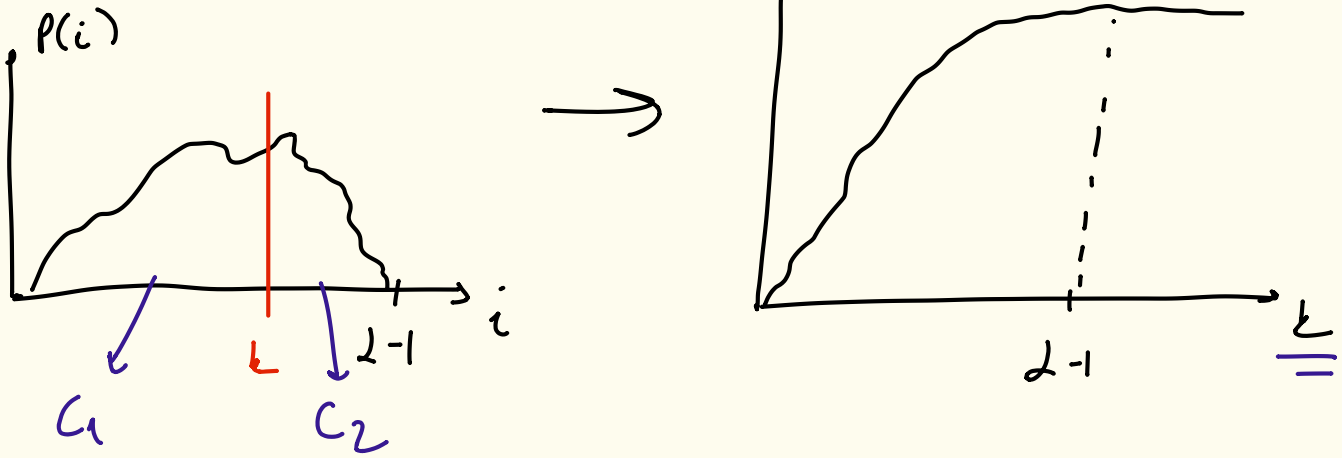
$P(1) = \frac{3}{5}$ $P(3) = \frac{4}{5}$

$P(2) = \frac{2}{5}$ $M.N = 9$

2-) Compute the cumulative sums. ($P_1(k)$ and $P_2(k)$)

$$P_1(k) = \sum_{i=0}^{k} p(i)$$

Total prob. of pixels in C1 (background) up to the threshold.



$k = 0 \dots L-1$

C1 threshold

$$P_2(k) = 1 - P_1(k)$$

C1 (foreground)

3-) Compute the cumulative means $m(k)$, $k = 0 \dots L-1$

Mean intensity of C1 (bg) " " C2 (fg)

$$\mu_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i \cdot p(i), \quad \mu_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i \cdot p(i)$$

using normalized histogram

The cumulative mean up to k is,

$$m(k) = \sum_{i=0}^{k} i \cdot p(i)$$

4-) Compute the **global mean** of the entire image.

$$m_G = \sum_{i=0}^{L-1} i \cdot p(i)$$

[We know

$$P_1 \cdot m_1 + P_2 \cdot m_2 = m_G$$

]

$L = [0, 0,255, 255)$

$m_G = (0, 0{,}5) + (255 \cdot 0{,}5) = 127{,}5$

5-) Compute the "**between class variance**". → How well bg and fg separated.

$$\sigma_B^2(t) = P_1(t) \cdot (m_1(t) - m_G)^2 + P_2(t) \cdot (m_2(t) - m_G)^2$$

$$\sigma_B^2(k) = \frac{(m_G \cdot P_1(k) - m(k))^2}{P_1(k) \cdot (1 - P_1(k))}$$

Aim is find k which maximize $\sigma_B^2(k)$ } The best threshold.

↳ k'nın her değeri için $\sigma_B^2(k)$ hesaplayacağız.

$P_1 = \frac{6}{10}$ $P_2 = \frac{4}{10}$

$m_1 = 20$ $m_2 = 225$

$mg = (10 \cdot \frac{2}{10}) + (20^2 \cdot \frac{2}{20}) + (30 \cdot \frac{2}{10}) + (220 \cdot \frac{2}{10})$

$+ (230 \cdot \frac{2}{10})$

$=$

$2 + 4 + 6 + 44 + 46$

$= (102)$

$\frac{6}{10} \cdot (20 - 102)^2 + \frac{4}{10} (225 - 102)^2$

~~1824+29~~

$\frac{6724.6}{10} + \frac{15129.4}{10} =$

fuy

$$[0, 0, 50, 50, 255, 255, 255, 100, 100, 100, 200, 200, 200]$$

Intensity (i)	Frequency (#i)	Probability ($p(i)$)
0	2	2/13 ≈ 0.1538
50	2	2/13 ≈ 0.1538
100	3	3/13 ≈ 0.2308
200	3	3/13 ≈ 0.2308
255	3	3/13 ≈ 0.2308

$$m_G = \sum_{i=0}^{255} i \cdot p(i)$$

Let's calculate:

$$m_G = (0 \times 0.1538) + (50 \times 0.1538) + (100 \times 0.2308) + (200 \times 0.2308) + (255 \times 0.2308)$$

$$m_G \approx 126.92$$

✅ **Threshold $k = 0$:**

1. **Group 1 (Background)**: Pixels $[0]$
   - $P_1(0) = 0.1538$
   - $m_1(0) = 0$
2. **Group 2 (Foreground)**: Pixels $[50, 100, 200, 255]$
   - $P_2(0) = 1 - P_1(0) = 0.8462$
   - $m_2(0) \approx 150$
3. **Global Mean** $m_G \approx 126.92$
4. **Between-Class Variance:**

$$\sigma_B^2(0) = 0.1538 \cdot (0 - 126.92)^2 + 0.8462 \cdot (150 - 126.92)^2$$

$$\sigma_B^2(0) \approx 1947.66$$

T = 50

C1 = [0, 50]

$P_1(50) = \frac{4}{13} = 0,3077$

$m_1(50) = \frac{0+0+50+50}{4} = 25$

C2 = [100, 200, 255]

$P_2(50) = \frac{9}{13} = 0,6923$

$m_2(50) = \ldots = 185,87$

$$\sigma_B^2(50) = 0{,}3077\,(25-126{,}52)^2 + 0{,}6923\,(185{,}83-126{,}52)^2$$

$$= 5341{,}52$$

⋮

Threshold k	Between-Class Variance $\sigma_B^2(k)$
0	1947.66
50	5341.92
100	**7212.46**
200	5907.18

→ optimal threshold.

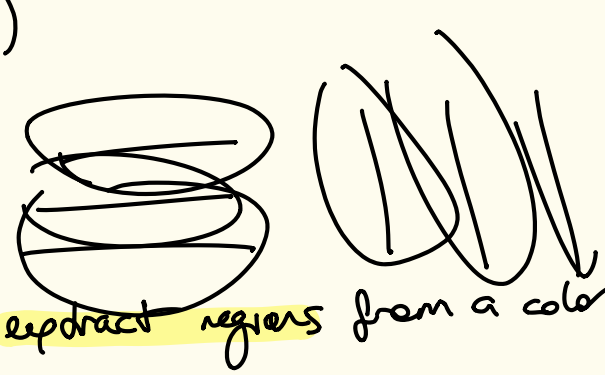
Thresholding in Color images

- Each pixel → (R, G, B)

$z = (z_1, z_2, z_3)$

↓

voxel



The goal is to extract regions from a color image that have a specified color.

Let a is the specified color.

$$g = \begin{cases} 1 & D(z,a) < T \quad \text{(pixel is part of the region)} \\ 0 & \text{otherwise} \end{cases}$$

g = output binary image

D(z,a) = Distance between the pixels color z and target color a.

$$D(z,a) = \|z-a\| = \left[(z-a)^T \cdot (z-a)\right]^{1/2} \rightarrow \text{Euclidean dist.}$$

- If $z = (100, 150, 200)$ and $a = (90, 140, 190)$, the Euclidean distance is calculated as:

$$D(z,a) = \sqrt{(100-90)^2 + (150-140)^2 + (200-190)^2}$$

$$D(z,a) = \left[(z-a)^T \cdot C^{-1} \cdot (z-a)\right]^{1/2} \rightarrow \text{Mahalanobis dist.}$$

C = Covariance matrix of the color dist.

Region splitting and Merging

Used for segment the image into regions based on criteria

Let R represent the image

Q: The predicate or condition that determines whether region satisfies the segmentation criteria.

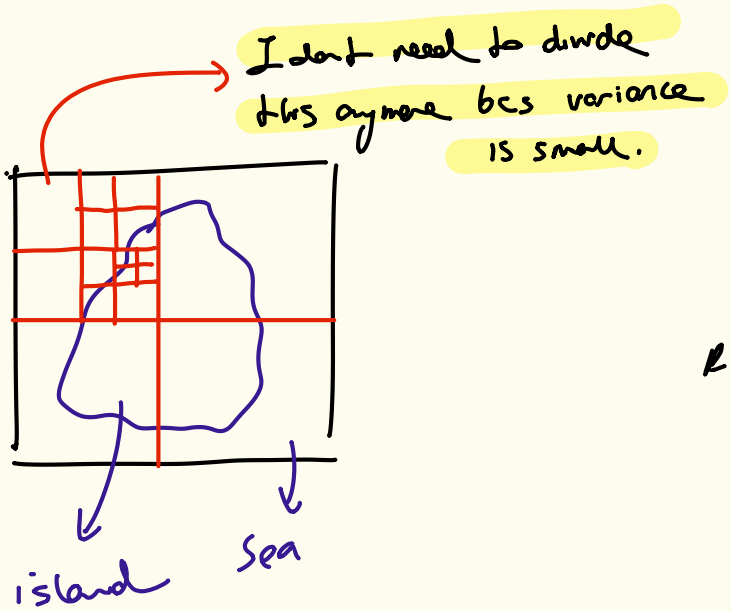
1-) Split the R into 4 quadrants

If $Q(R_i)$ = false, split the region further into smaller quadrants.

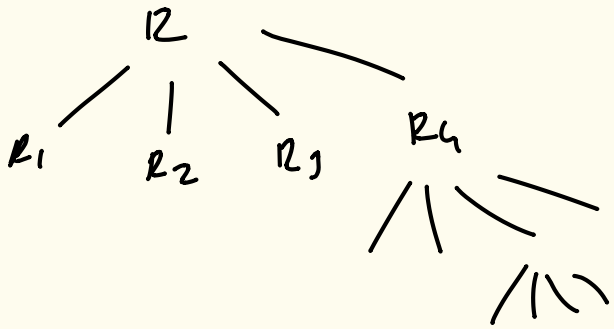
Keep splitting until $Q(R_i)$ = true (region satisfies the condition)

2-) When no splitting is possible, merge any adjacent regions R_j and R_k where $Q(R_j \cup R_k)$ = true.

3-) Stop when no further merging is possible

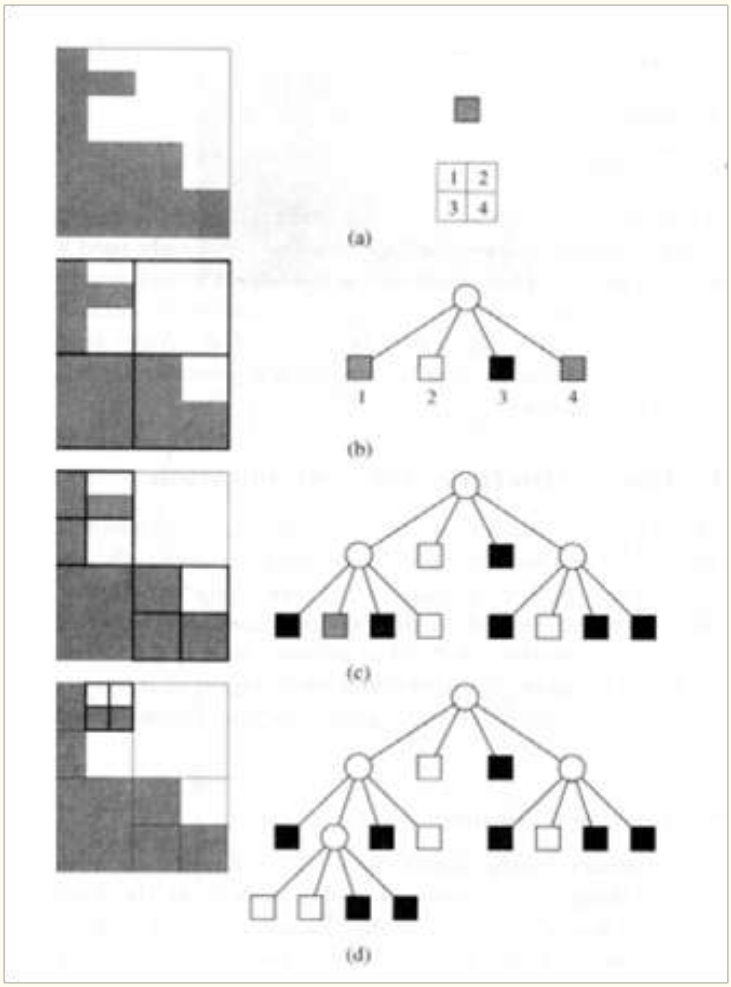


Then prune the images.!



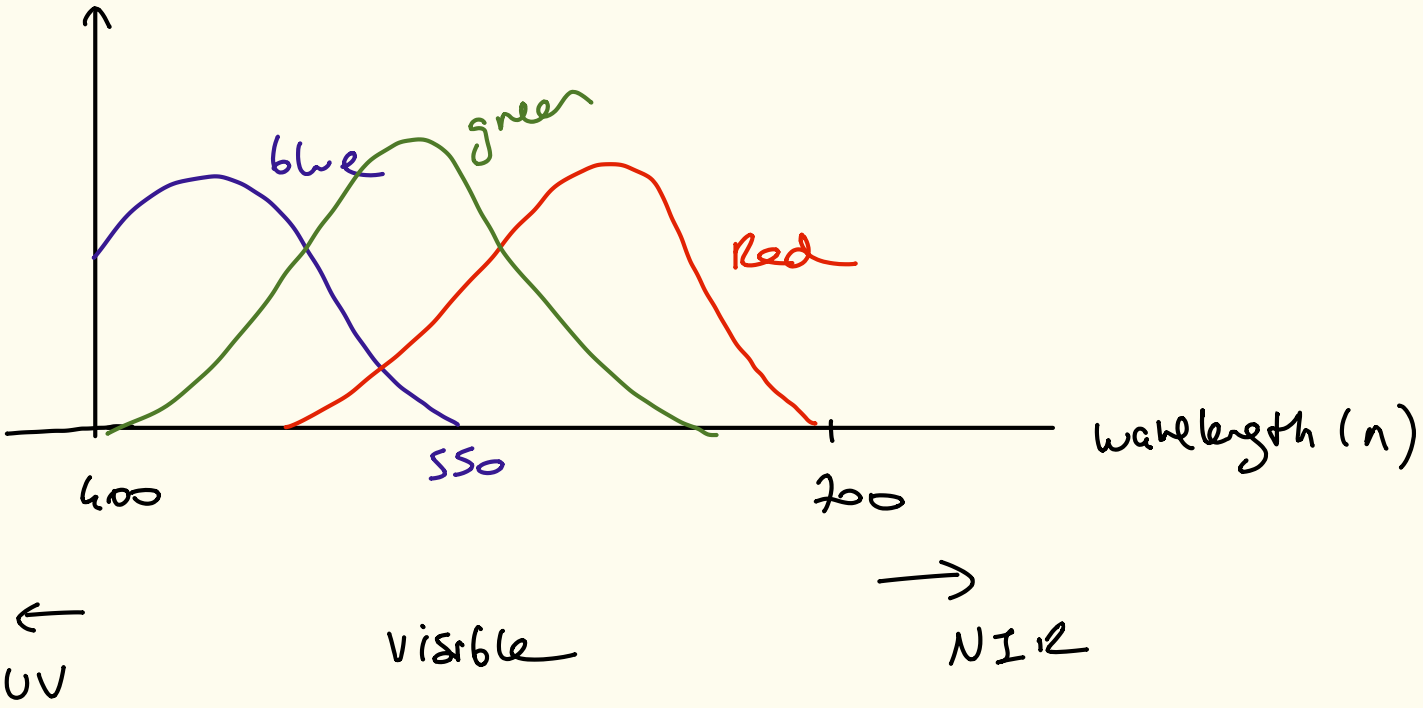
e.g.

Q → Could be variance or intensity range



Color Processing

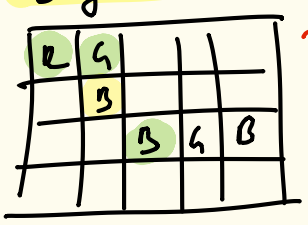
We have 7 million cones (%65 sensitive to Red
%33 " " green
%2 " " blue)



Color Models

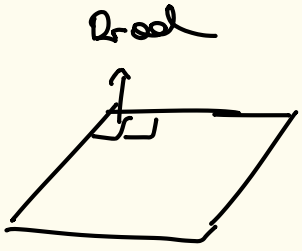
RGB for color cameras

Bayern Pattern



komşuların değerlerine bakarak kendi değerini tahmin ediyorum.

"neighbors are interpolated to estimate missing colors."

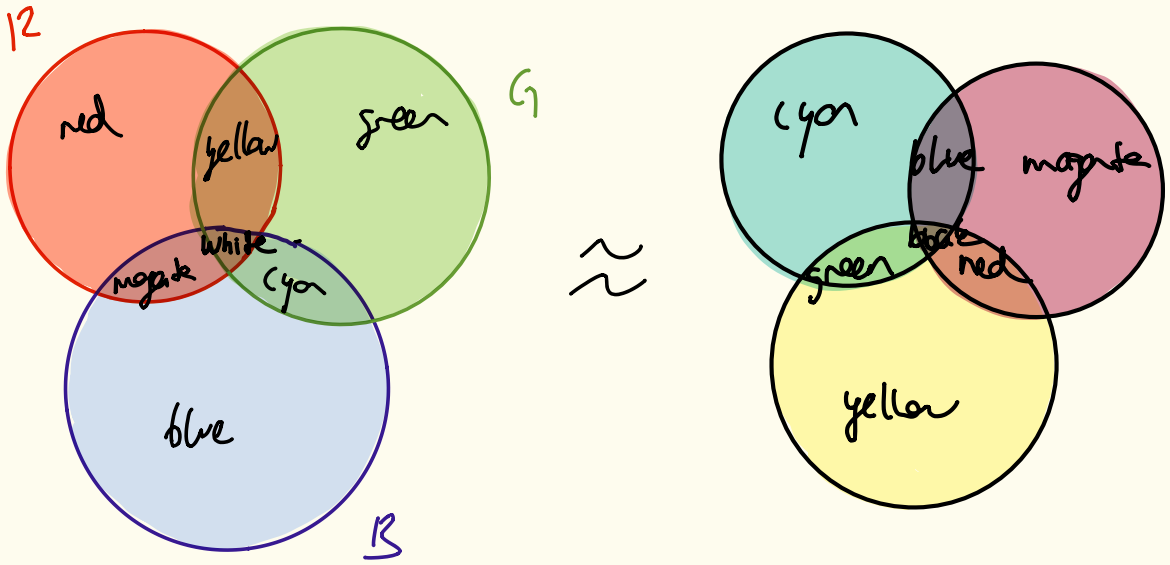


MxN

↳ DC 2 AC

CMY and CMYK (cyan, magenta, yellow, black)

RGB



red, green, blue

↓

primary colors

C, M, Y (k)

↓

primary pigment

- Magenta pigment absorb green light and reflect remaining (red, blue)
- Cyan " " red " " "
- Yellow " " blue " " "

* RGB -> Primary colors of light.

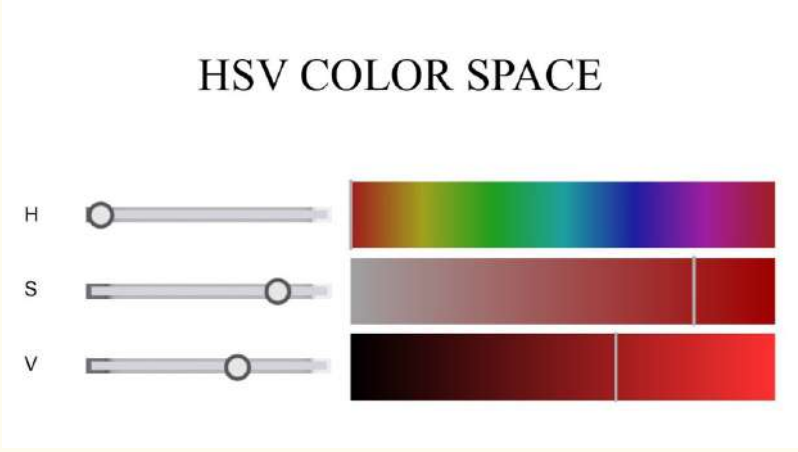
* CMYK -> " " pigments.

Other color models:

* HSI (Hue, Saturation, Intensity)
  - Saturation: ↓ pure vs vivid
  - Intensity: ↓ brightness

* HSV (Hue, Saturation, Value)
  - Value: ↓ brightness

* Lab



BAŞLIK

HSI

Hue (renk özü) → Describes the pure color.

Saturation (Doygunluk) → Refers to the relative humidity or the amount of white light mixed with hue.

Intensity → Intensity of the light.

RGB to HSI Conversion

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases}$$

For $B \leq G$:

$$H = \cos^{-1}\left(\frac{1/2\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right)$$

For $B > G$:

$$H = 360° - H$$

$$S = 1 - \frac{3}{R+G+B} \cdot \min(R,G,B)$$

$$I = \frac{1}{3}(R+G+B)$$

HSV → Similar to HSI but $V = \max(R,G,B)$

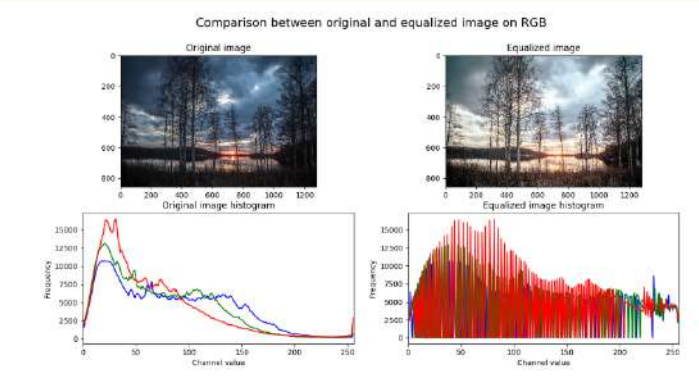
Color Histogram Processing } To enhance the contrast and brightness.

RGB → I → Apply histogram processing to enhance brightness and contrast.

S → keep " "

H → " "

Adjusting intensity affects how dark or light the image appears.



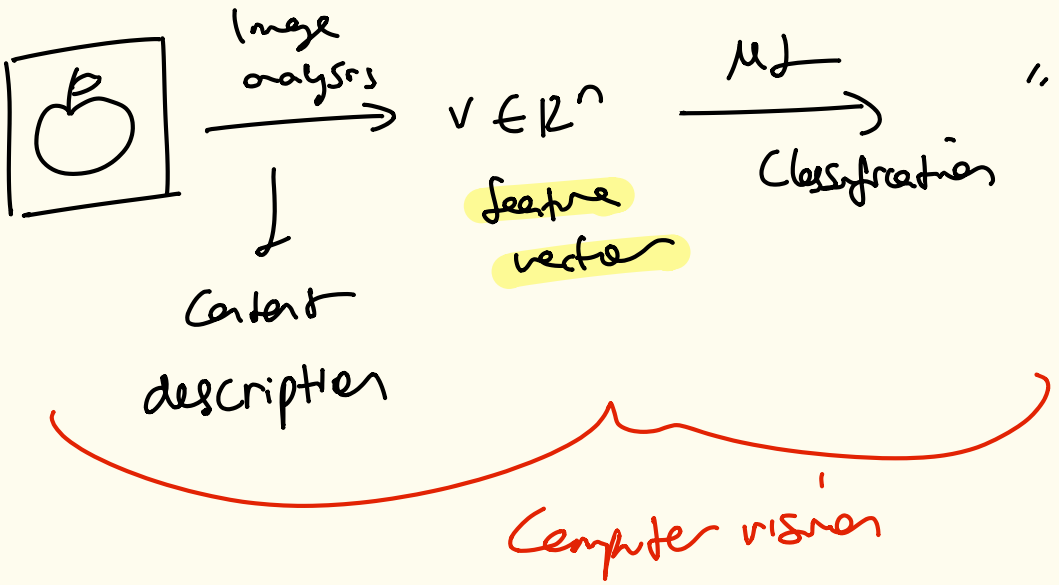
Color Image Sharpening

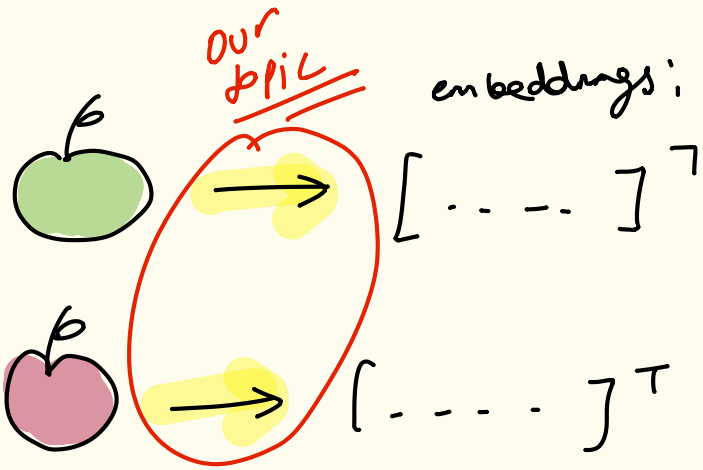
1. Apply Laplacian to each RGB channel:

$$\nabla^2(C(x,y)) = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix}$$

2. OR, convert the image to HSI and sharpen the intensity (I) (Laplacian)

Image Description





∝ To get nice result from ML algos, we should describe image better.

Problems in Image Description

? ! ∝ High intra-class variance (red and green apple)

∝ low inter-class " (red apple and red cherry)

∝ Robust to: illumination (light condition), temperature (warmer or cooler tones), direction (viewpoint)

∝ Rotation / Viewpoint

∝ Occulusion (Part of object hidden)

∝ Deformations (face recognition)

∝ Scale (distance) variations (Some object can have different sizes in image)

→ That why image description is difficult.

Solutions:

Increase quantization color ↑ more colors ↑

∝ Reduces # of colors in image by grouping similer ones.

quantization → k-means clustering

Clusters will be your palette colors.

! ? You losing some details but you keep the image

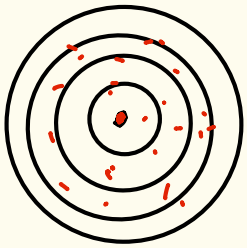
Color auto-correlogram

Measures how spatially connected similar colors are in image.

- In an image of a red apple, a **color autocorrelogram** will check how **connected the red pixels** are across the image.
- If the red pixels are **spread out** uniformly, it indicates a large object.
- If they are **clustered**, it indicates a smaller object.

Color distribution entropy:

- kırmızı pikselların merkezi



entropi.

- If **red pixels** are **concentrated in one area**, entropy is low.
- If **red pixels** are **spread across the image**, entropy is high.

Help identifying color patterns