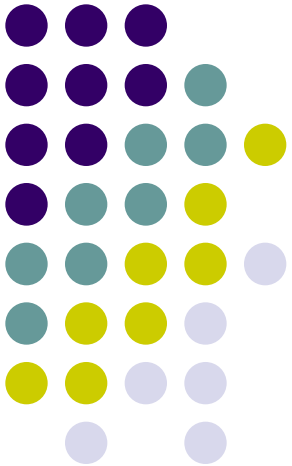
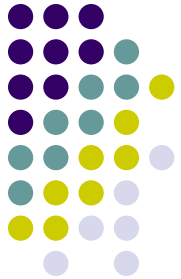
# İMGE İŞLEME
# Ders-2

İmge Dosya Tipleri ve Temel İşlemler

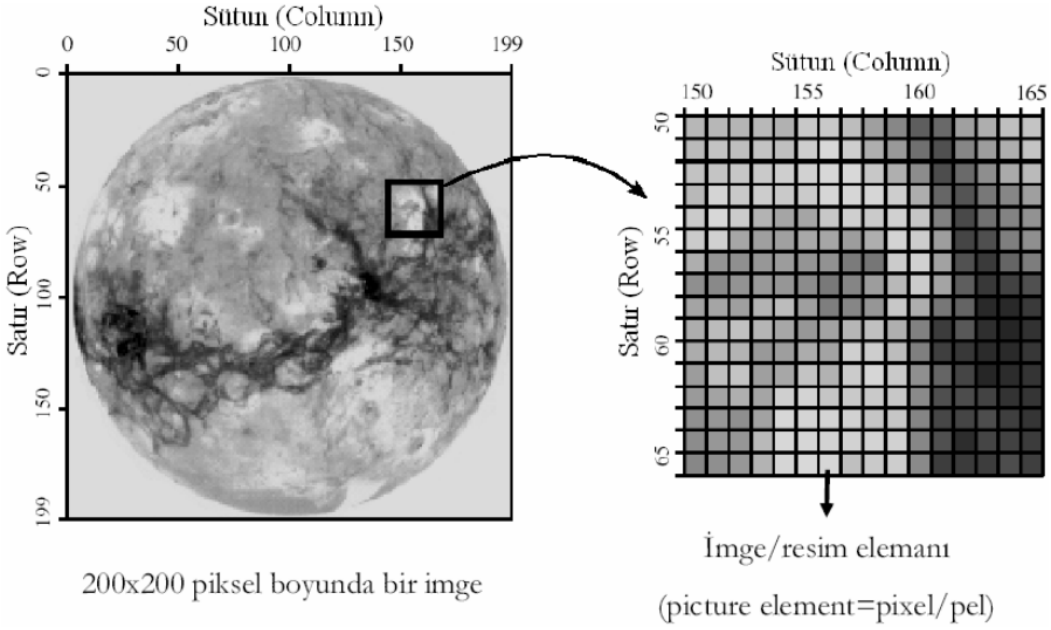



(Yrd. Doç. Dr. M. Kemal GÜLLÜ)

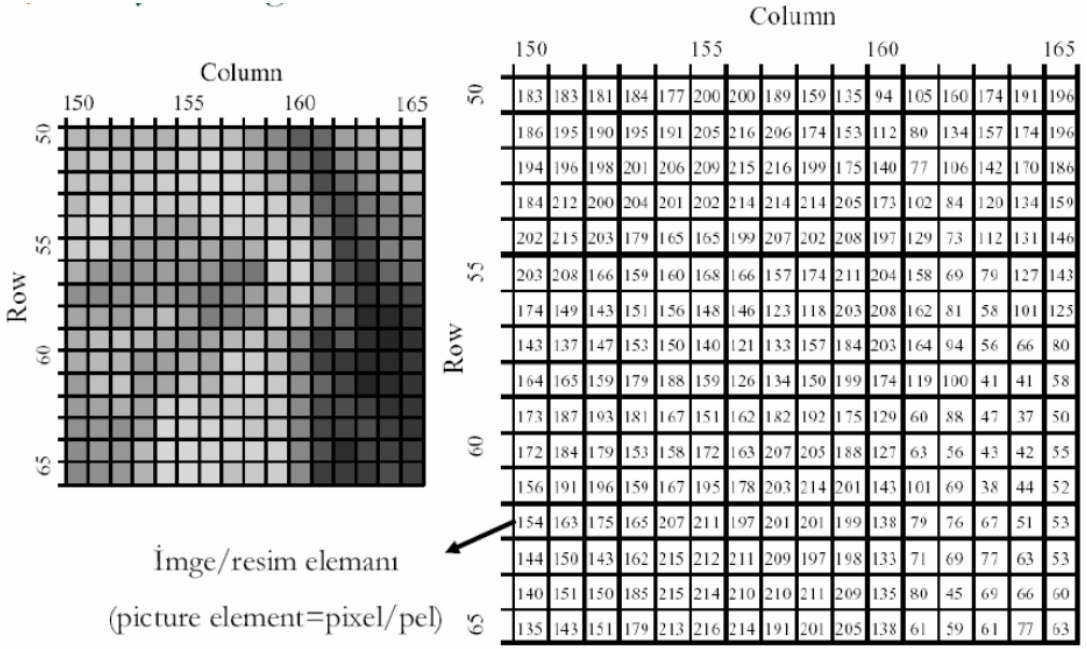
Dersin web sayfası: http://mf.kou.edu.tr/elohab/kemalg/imge_web/odev.htm



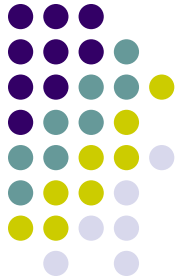
# MATLAB temel bilgiler




200x200 piksel boyunda bir imge



# İmge Dosya Tipleri



Sayısal imgeler genellikle bmp, jpg, tiff, raw gibi formatlarda saklanmaktadır.

Gri tonlu, 8bit/piksel bit derinliğinde, 1000x1000 piksel boyutlu bir imgenin bellekte kaplayacağı alan nedir?

Yanıt: 1000000 bayt = 977 kbayt

Eğer yukarıdaki imge renkli olsaydı, bu imgenin bellekte kaplayacağı alan:

977 kbayt x 3 = 2931 kbayt = 2.86 Mbayt

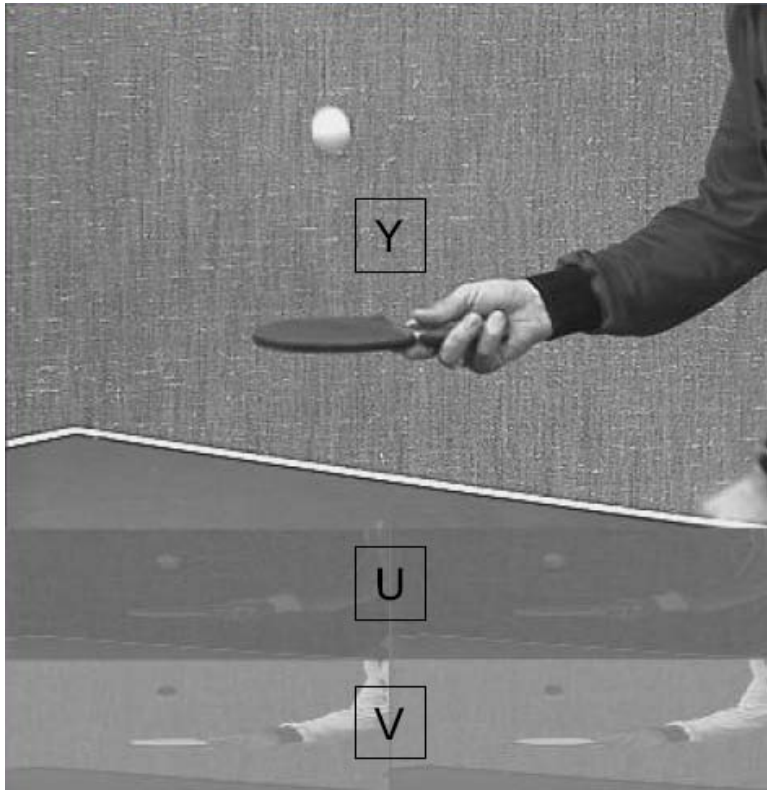
# İmge Dosya Tipleri (raw)

- Yalnızca piksel ışıklılık değerlerini barındıran dosya tipidir.
- İmgenin piksel boyutunu gösteren herhangi bir başlık bilgisi içermez.
- İmgeyi açmal için piksel boyutunu bilmek gerekir.
- Bu tip dosyaları Matlab ya da C gibi programlama dillerini kullanarak açmak için bilinen dosya açma işlemlerini yapmak gerekmektedir.

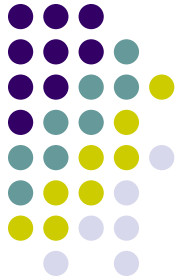
```
w=256;          % imgenin yatay boyutu
h=256;          % imgenin düşey boyutu
f=fopen('C:\Documents\lena.raw','r');
% açılacak dosyanın konumu ve adı f dosya değişkenine yüklendi

I=fread(f);     % imge I değişkenine dizi olarak alındı
I=reshape(I,w,h);       % 2-boyutlu matrise dönüştürüldü
status=fclose(f);       % dosya kapandı

figure; imshow(uint8(I'));      % Ekranda imge görüntülendi
```

# İmge Dosya Tipleri (yuv)

- Renk bileşenleri sıkıştırılmıştır.
- Raw dosya tipine benzer şekilde, bu dosya tipinde de imgenin piksel boyutu dosya içerisinde yoktur.
- Bu nedenle imgenin boyutlarının önceden bilinmesi gerekmektedir.



# İmge Dosya Tipleri (bmp)



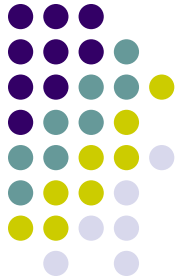
- Sıkça kullanılan bir imge dosya tipidir.
- Sıkıştırılmamış ve sıkıştırılmış dosya yapısı mevcuttur.
- Sıkıştırılmamış dosya yapısında, raw dosya tipine ek olarak imgenin piksel boyutu, bit derinliği gibi bilgileri turan başlık kısmı vardır.
- Matlab ile .bmp uzatılı bir imgeyi okumak için

```
I=imread('C:\Documents\lena.bmp');
```

Kodunu yazmak yeterlidir.

- C ve diğer programlama dillerinde ise mevcut kütüphanelerden faydalanılarak bu dosya tipi okunabilmektedir.

# İmge Dosya Tipleri (diğer)



- Tiff, jpeg, png, gif, pbm, pgm, hdf, pcx... gibi imge dosya tipleri de vardır.

- Tiff genellikle bmp benzeri bir dosya tipi olmakta birlikte kayıplı sıkıştırma modu da vardır.
- Bunun yanında 10 bit/piksel, 16 bit/piksel gibi bit derinliklerinde kayıt olanagına sahiptir.
- Matlab “imread” komutu ile okunmaktadır.

- Jpeg kayıplı bir imge dosya tipidir.
- Bu sıkıştırma kullanılarak imge yüksek verimlilikle sıkıştırılabilmektedir.
- Bu dosya tipi de Matlab “imread” komutu ile okunmaktadır.

# İmge standart boyutları

- CIF: Common Intermediate Format
  - VGA: Video Graphics Array
- SIF: Source Intermediate Format

| | |
|---|---|
| SQCIF | 128 × 96 |
| QCIF | 176 × 144 |
| CIF | 352 × 288 |
| 4CIF | 704 × 576 |
| 16CIF | 1408 × 1152 |
| VGA | 640 × 480 |
| QVGA | 320 × 240 |
| SCIF | 352 × 240 |

# devirme

$$B = A'$$

$$B(j, i) = A(i, j)$$

$$(i = 0, \ldots, N-1,\ j = 0, \ldots, M-1)$$

# düşeyde çevirme

$$B\left(i, M-j-1\right) = A(i, j)$$

$$\left(i = 0,\ldots, N-1\ , j = 0,\ldots, M-1\right)$$

# döndürme

- 90°, 180 °, 270 ° gibi açılarda döndürme işlemlerini kolayca gerçekleştirebiliriz.
- Bu açıların dışındaki değerlerde ise açısal döndürme işlemlerinin (Sin x, Cos x değerlerini kullanarak) yapılması gerekmektedir.
- Bunun yerine, Matlab hazır işlevlerinden "imrotate" kullanılabilir.

**Ir=imrotate(I,açı,yöntem);**

açı: saat yönünün tersi dönülecek açı değeri.

yöntem: döndürme işlemi sonrasında yeni piksel değerlerinin hesaplanacağı aradeğerleme yöntemi.

'nearest', 'bilinear', 'bicubic',

Örn;

```
Ir=imrotate(I,45, 'bilinear');
```

# kırpma

$$B(i,j) = A(n_1 + i, n_2 + j)$$

$$(i = 0,\ldots,m_1 - 1\ ,\ j = 0,\ldots,m_2 - 1)$$

$(n_1, n_2) \rightarrow$ sol üst köşe noktası

$(m_1, m_2) \rightarrow$ pencere boyutları

# öteleme

$$B(i, j) = A(i - n_1 + 1, j - n_2 + 1)$$

$$(i = n_1, ..., N, j = n_2, ..., M)$$

$(n_1, n_2)$ sıfır noktası

# öteleme

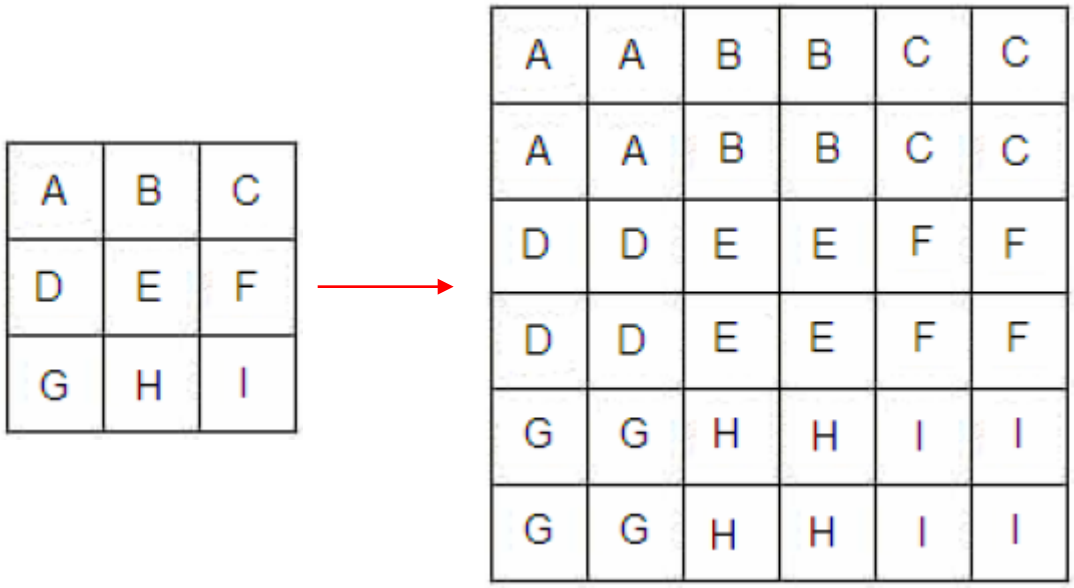
Öteleme işlemi yapan bir Matlab işlevi yazalım:

```
function [B]=my_otele(A,n1,n2)

[w,h]=size(A);

B=zeros(w,h);

for i=n1:w
   for j=n2:h

      B(i,j)=A(i-n1+1,j-n2+1);

   end
end
```

Burada for döngüleri yerine tek bir satır yazarak aynı işlem yapılabilir.

# boyut değiştirme-yakınlaştırma

- Yakınlaştırma, düşük piksel boyutlu bir imgenin piksel boyutunun yazılımsal olarak arttırılmasıdır.
- Sayısal yakınlaştırma (digital zoom).



# boyut değiştirme-yakınlaştırma

- Boyut büyültmede daha yumuşak geçişler için:

| | | |
|---|---|---|
| A | B | C |
| D | E | F |
| G | H | I |



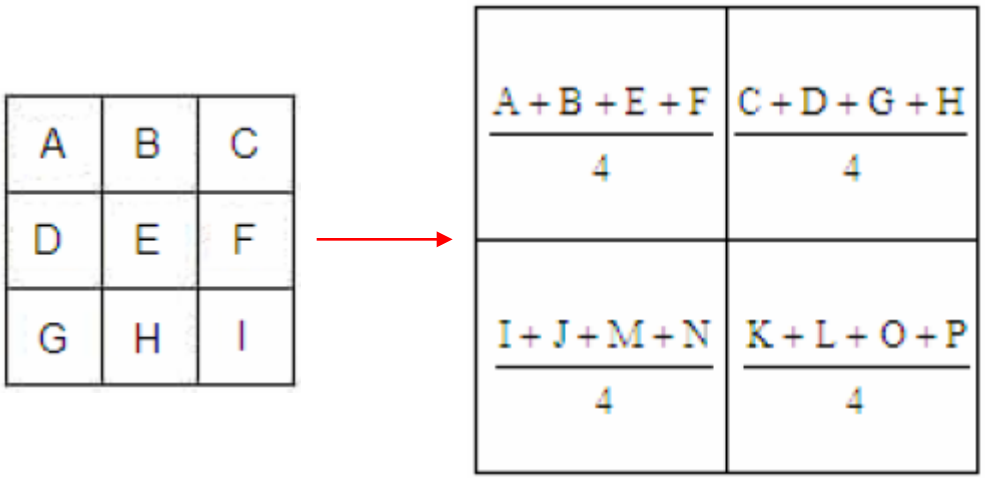
| | | | | |
|---|---|---|---|---|
| A | $\frac{A+B}{2}$ | B | $\frac{B+C}{2}$ | C |
| $\frac{A+D}{2}$ | $\frac{A+B+D+E}{4}$ | $\frac{B+E}{2}$ | $\frac{B+C+E+F}{4}$ | $\frac{C+F}{2}$ |
| D | $\frac{D+E}{2}$ | E | $\frac{E+F}{2}$ | F |
| $\frac{D+G}{2}$ | $\frac{D+E+G+H}{4}$ | $\frac{E+H}{2}$ | $\frac{E+F+H+I}{4}$ | $\frac{F+I}{2}$ |
| G | $\frac{G+H}{2}$ | H | $\frac{H+I}{2}$ | I |

# boyut değiştirme-yakınlaştırma

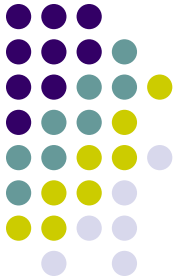
- Hangisi daha görünür?

# boyut değiştirme-uzaklaştırma

- Birden fazla pikselin değeri çeşitli matematiksel işlemlerden geçirilerek bir piksele atanır.



# boyut değiştirme



- Matlab ile boyut değiştirme için “imresize” adındaki işlev kullanılabilmektedir.

```
Is=imresize(I,oran,yöntem);
```

oran : giriş imgesinin boyutunun değişme oranını verir. oran>1 (büyütme), oran<1 (küçültme).

yöntem : boyut değiştirmede kullanılacak aradeğerleme yöntemi.

Örn;

```
Is=imresize(I,0.97, ‘bicubic’);
```

# İmge oluşturma

$$\mathbf{A} = \left[\begin{array}{ccccc} 0 & 1 & 2 & \dots & 255 \\ 0 & 1 & 2 & \dots & 255 \\ \vdots & & & & \\ 0 & 1 & 2 & \dots & 255 \end{array}\right] \Bigg\} 256 \text{ rows}$$

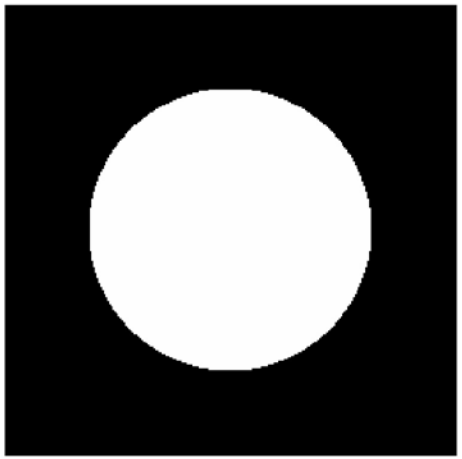
256

256

```
for i = 1 : 256
  for j = 1 : 256
    A(i, j) = j − 1;
    end
  end
```

# İmge oluşturma

- (128,128) merkezli, yarıçapı 80 piksel beyaz bir daire

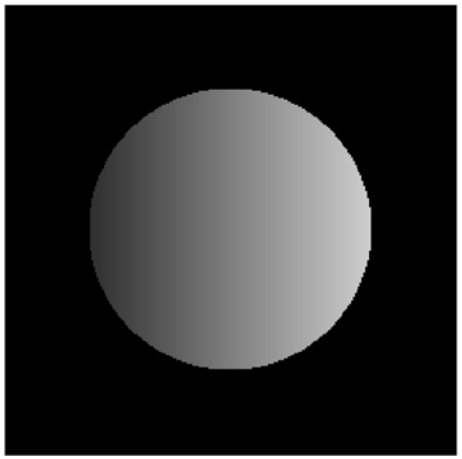


$$B(i,j) = \begin{cases} 255 & \text{if } \sqrt{(i-128)^2 + (j-128)^2} < 80 \\ 0 & \text{otherwise} \end{cases}$$

```
for i = 1 : 256
  for j = 1 : 256
    dist = ((i − 128)^2 + (j − 128)^2)^(.5);
    if (dist < 80)
      B(i, j) = 255;
    else
      B(i, j) = 0;
      end
    end
  end
```

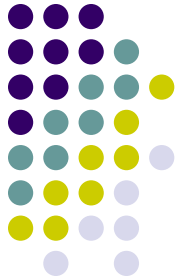
# İmge oluşturma

- ???




```
for i = 1 : 256
  for j = 1 : 256
    C(i, j) = A(i, j) * B(i, j)/255;
    end
  end
```

# Ortalama ve Değişinti



- Bir imgenin örnek ortalaması (sample mean):

$$m_A = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i,j)}{NM}$$

- Örnek değişintisi (sample variance):

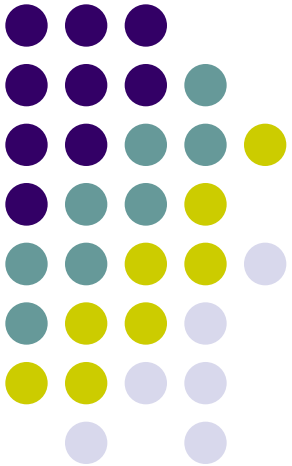
$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i,j) - m_A)^2}{NM}$$

- Örnek standart sapması (örnek st. sapma):

$$\sigma_A = \sqrt{\sigma_A^2}$$

# İMGE İŞLEME
# Ders-3

İmge Pekiştirme
(Nokta İşlemleri)




(Yrd. Doç. Dr. M. Kemal GÜLLÜ)

Dersin web sayfası: http://mf.kou.edu.tr/elohab/kemalg/imge_web/odev.htm

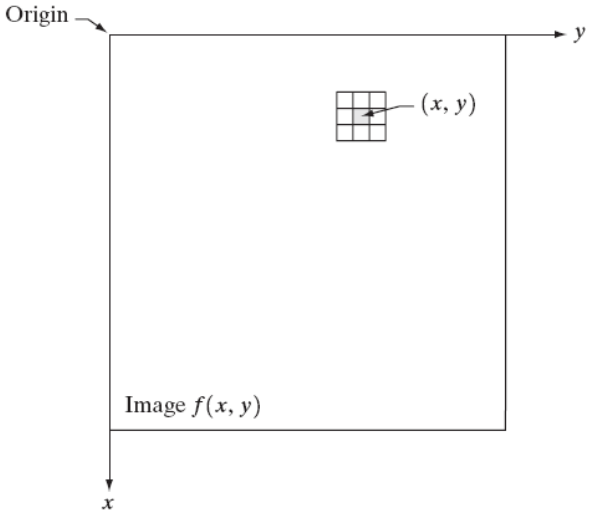
# Nokta İşlemleri

- Piksellerden oluşan imge uzayına uzamsal düzlem (spatial domain) denir.
- Uzamsal düzlem işlemleri aşağıdaki gösterimle ifade edilmektedir.

$$g(x,y) = T\left[f(x,y)\right]$$

işlev

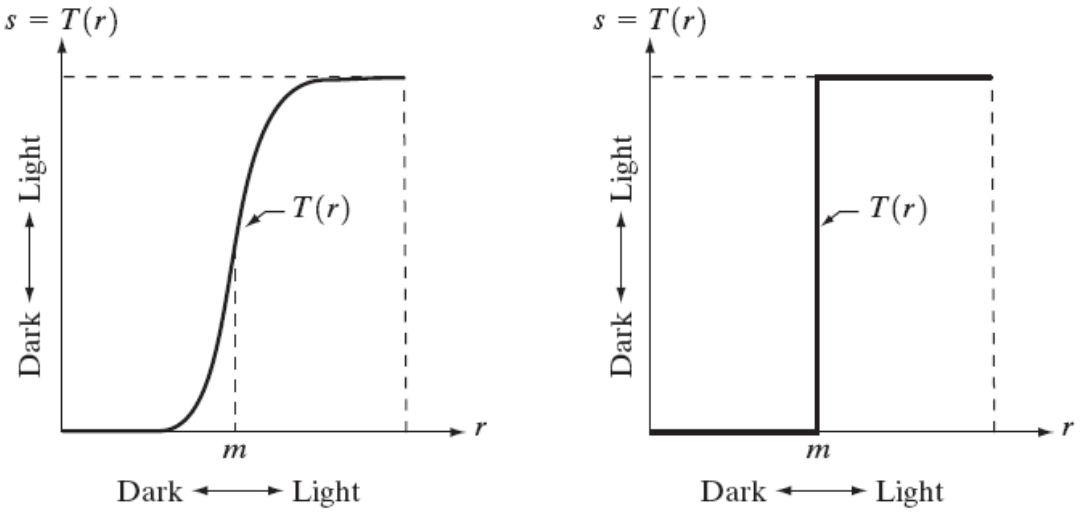
- Buradaki T işlevi, doğrudan (x,y) pikselini işleyebileceği gibi, (x,y) pikselinin komşuluklarını da hesaba katabilir.



# Nokta İşlemleri

- Eğer komşuluk boyutu 1x1 ise (yalnızca (x,y) pikseli alınıyor), bu durumda T *gri-seviye dönüşüm işlevi (grayscale-level transformation function)* olarak adlandırılır.
- Bu tür işlemlere de *nokta işlemleri (point operations)* adı verilir.
- Bu işlem kısaca aşağıdaki şekilde yazılabilir.

$$s = T(r)$$



# Parlaklık Ayarı

$$g(x,y) = T\left[ f(x,y) \right]$$
$$= f(x,y) + b$$

b>0 ise parlaklık artar

b<0 ise parlaklık azalır

$$s = r + b$$

orjinal

b = -50

b = +50

# Karşıtlık (Kontrast) Ayarı

$$g(x,y) = T\left[ f(x,y) \right]$$
$$= af(x,y)$$

a>1 ise karşıtlık artar

a<1 ise karşıtlık azalır

$$s = ar$$

orjinal

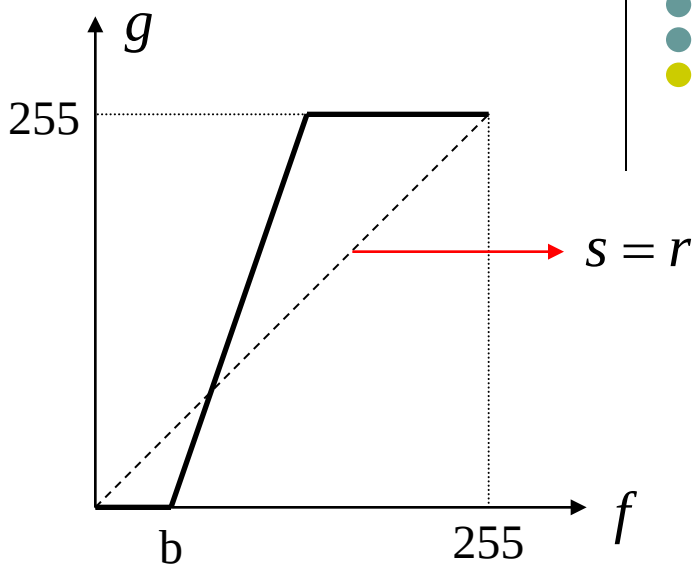
a = 0.5

a = 2

# Parlaklık+Karşıtlık Ayarı

$$g(x,y) = T\left[ f(x,y) \right]$$
$$= af(x,y) + b$$

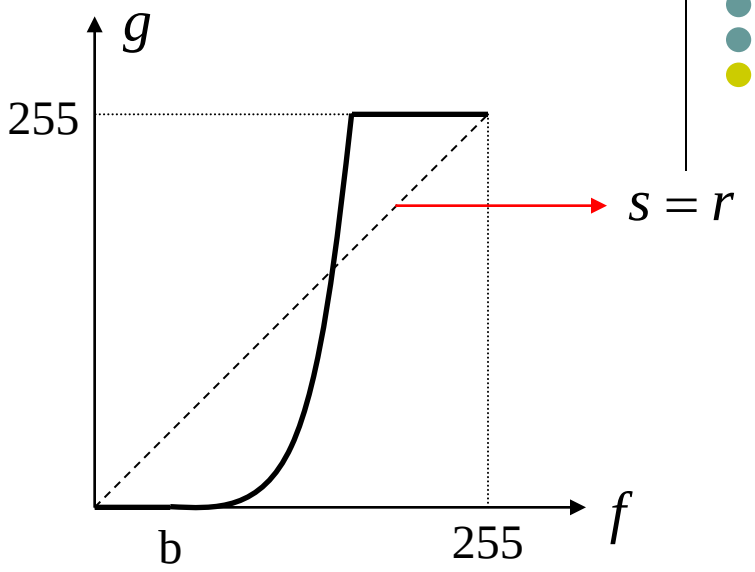
$$s = ar + b$$



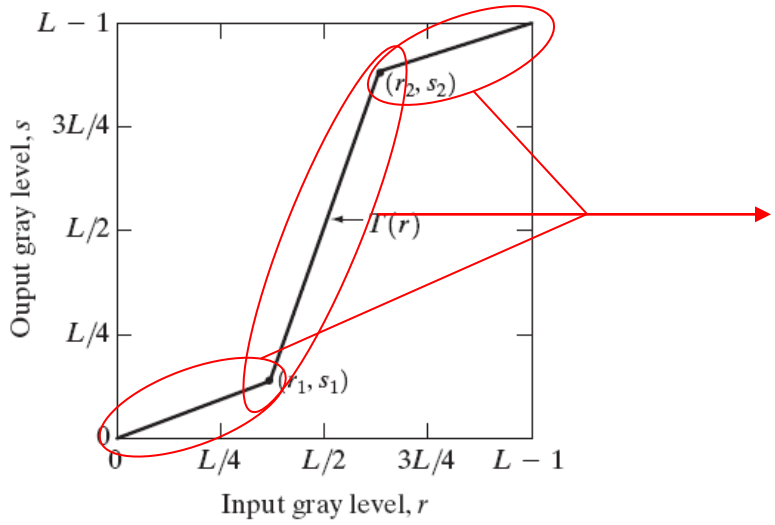

# Parlaklık+Karşıtlık Ayarı

$$g(x,y) = T\left[ f(x,y) \right]$$

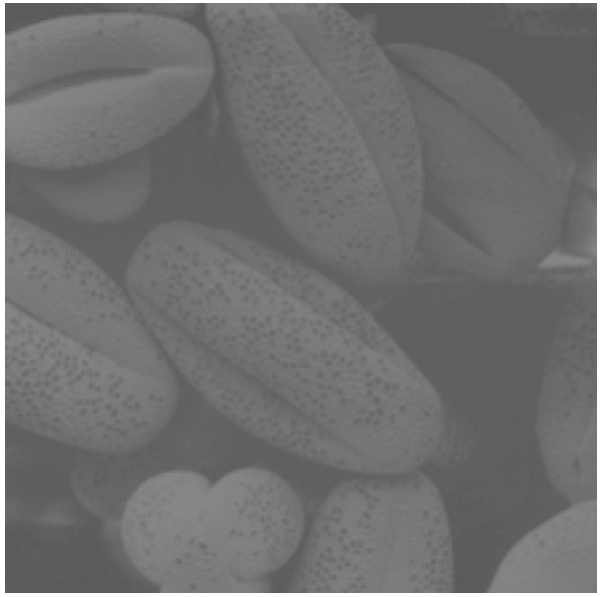
MATLAB **imadjust** işlevi

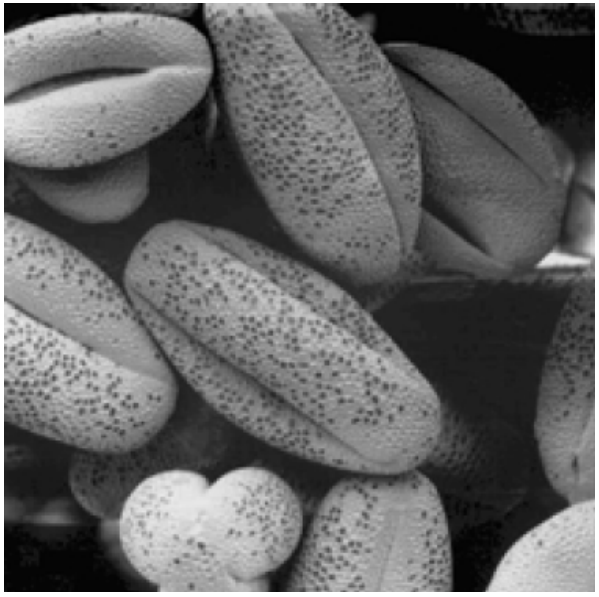



# Parlaklık+Karşıtlık Ayarı

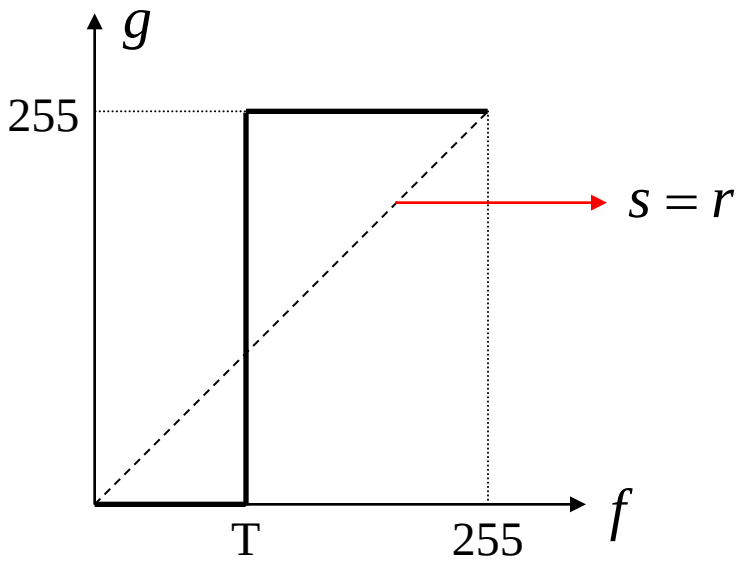



Kısmi-doğrusal dönüşüm



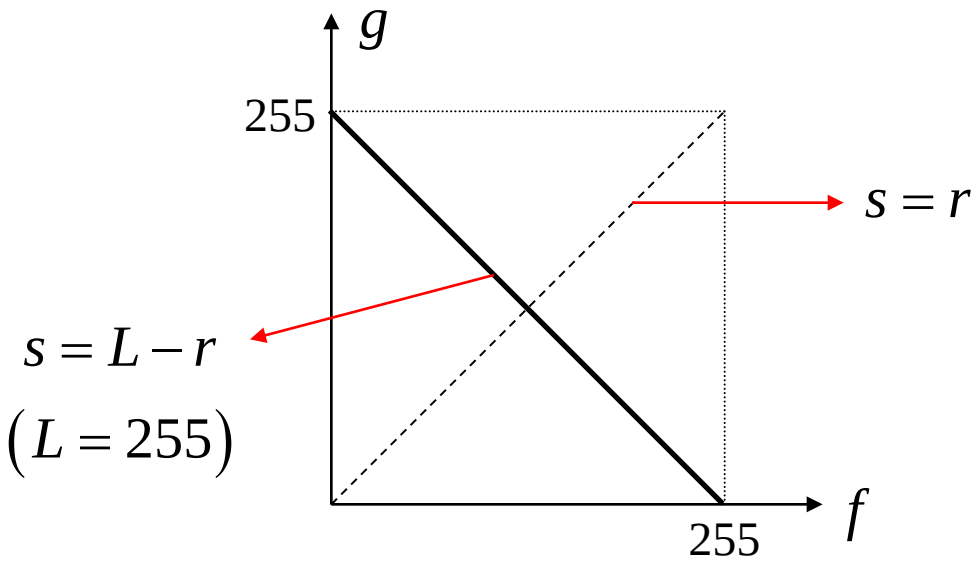


# Eşikleme




Sonuçta ikili (binary) imge oluşuyor.

# Olumsuzlama




# Histogram

- Her bir gri ton seviyesinin ([0,255]) imgedeki bulunma sıklığını (frekansını) gösterir.
- Yani imgedeki piksellerin dağılımı hakkında bilgi verir.
- İmge pekiştirmede sıkça kullanılmaktadır.

$$h(r_k) = n_k$$

$r_k$ : $k$. gri seviye

$n_k$ : $k$. gri seviyedeki toplam piksel sayısı

- Histogram normalize edildiğinde ise gri seviyelerin imge içerisindeki bulunma olasılıklarını verir.

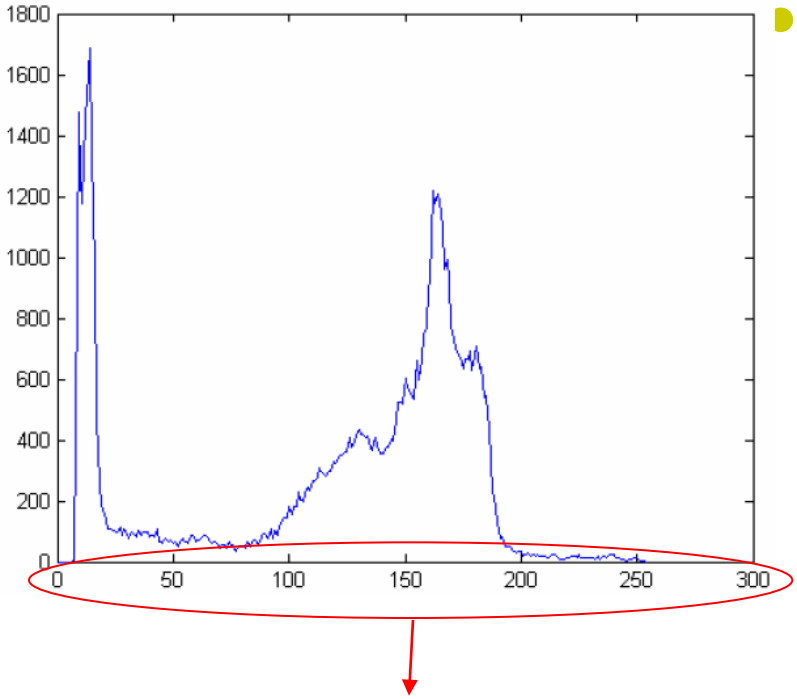
İlgili seviyenini olasılık değeri

$$p(r_k) = n_k / n$$

imgedeki toplam piksel sayısı

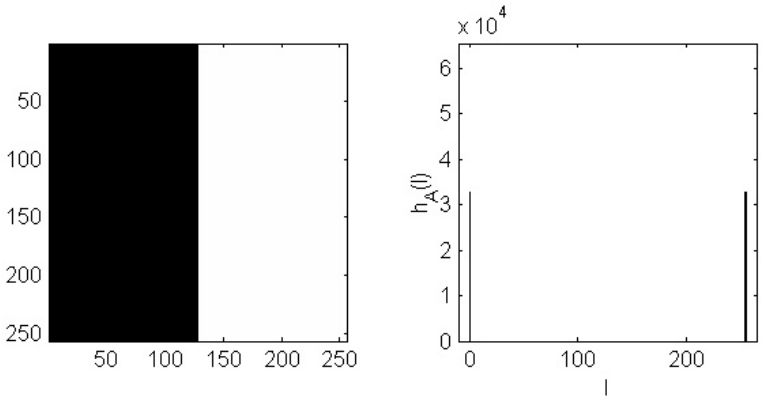
$$k = 0,1,\ldots,L-1$$

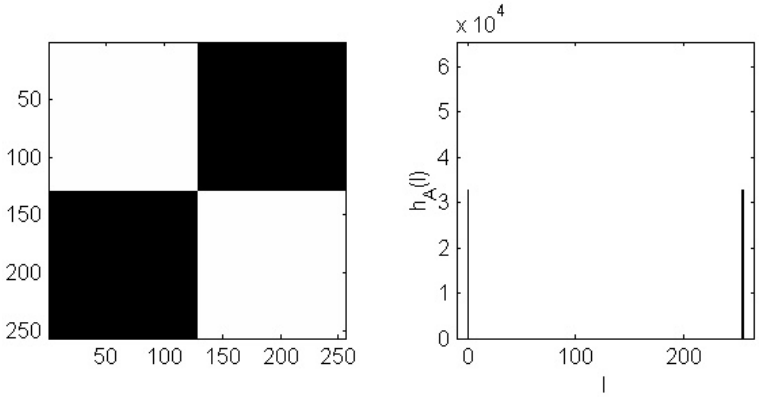
# Histogram




MATLAB **imhist** işlevi

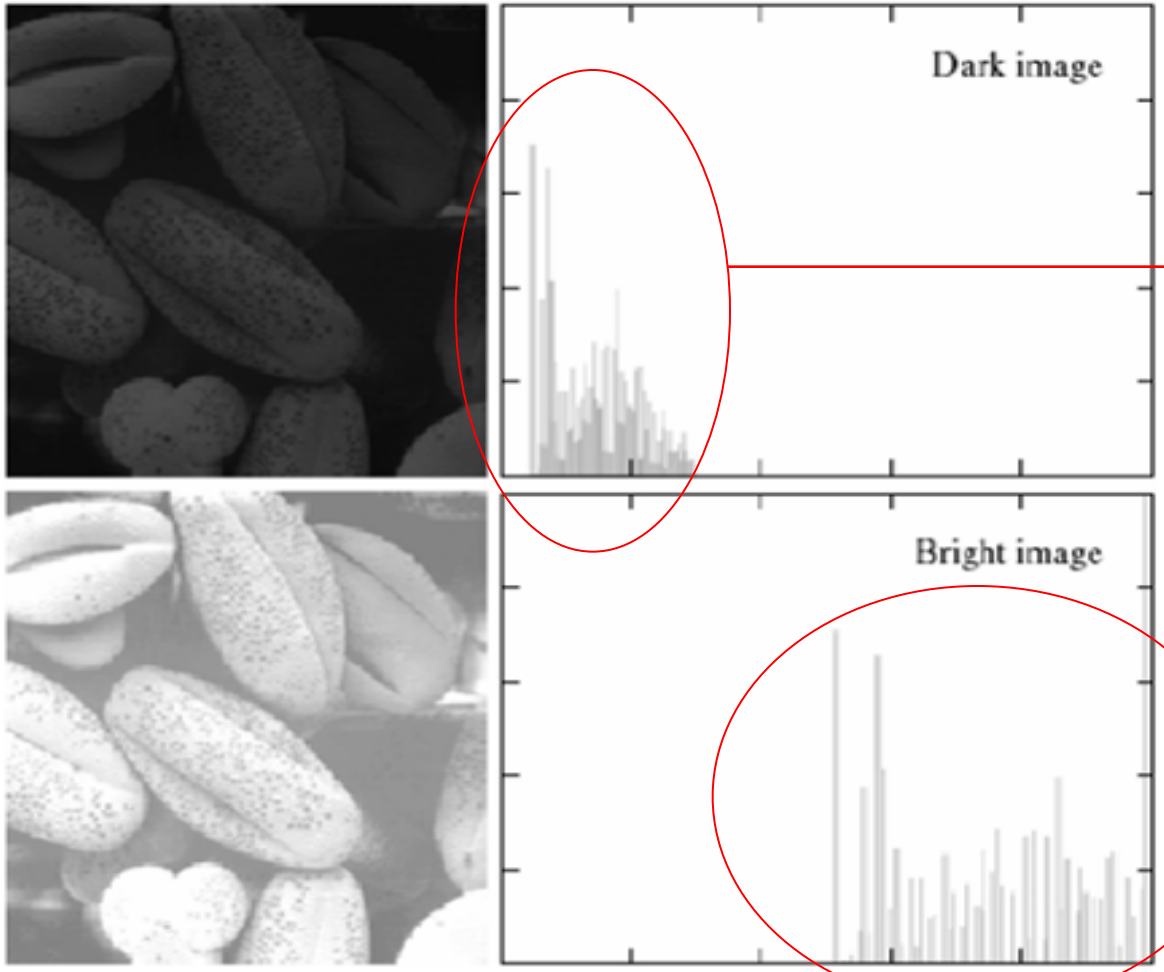
# Histogram




Piksel konum bilgisi bulunmaz!

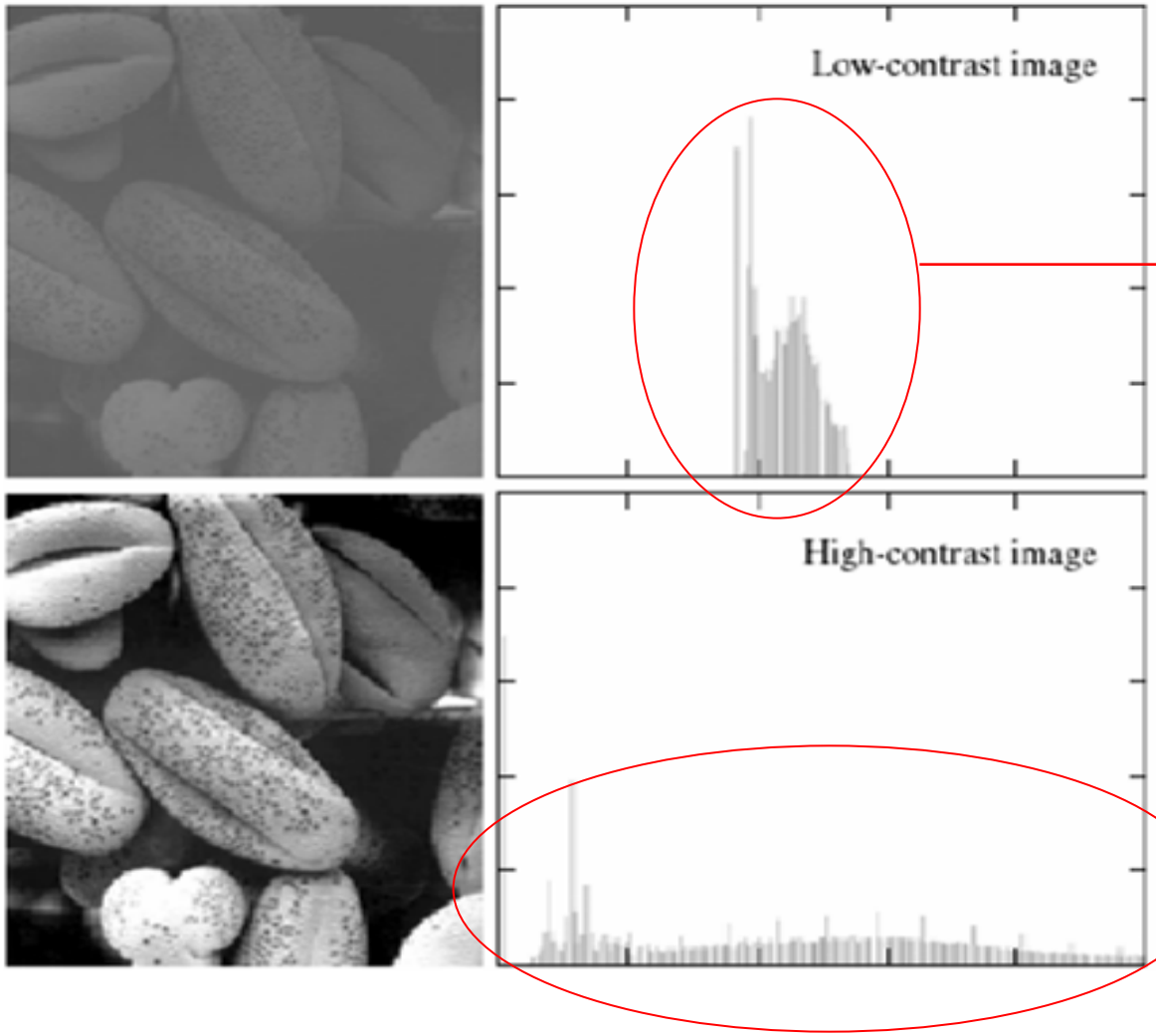
# Histogram




Karanlık imge

Parlak imge

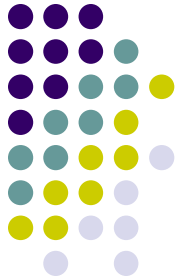
# Histogram




Karşıtlığı düşük imge

Karşıtlığı yüksek imge

# Sürekli Genlik Rastlantı Değişkenleri



- Let $\chi$ be a continuous amplitude random variable $\chi \in (-\infty, +\infty)$.

  $f_\chi(x)$: the probability density function of $\chi$,
  $F_\chi(x)$: the probability distribution function of $\chi$.

$$f_\chi(x)dx = \text{Probability}(x \leq \chi < x + dx)$$
$$F_\chi(x) = \text{Probability}(\chi \leq x)$$

- Properties:

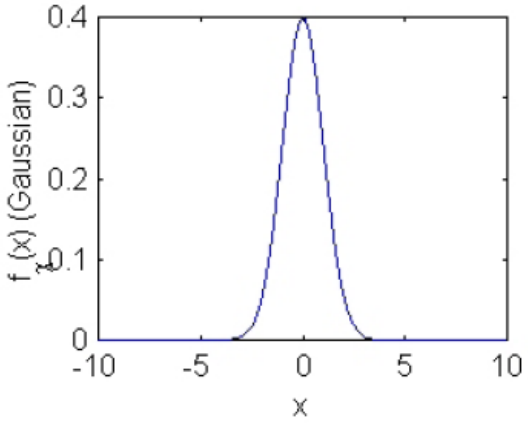
$$F_\chi(x) = \int_{-\infty}^{x} f_\chi(t)dt \Rightarrow \frac{dF_\chi(x)}{dx} = f_\chi(x)$$

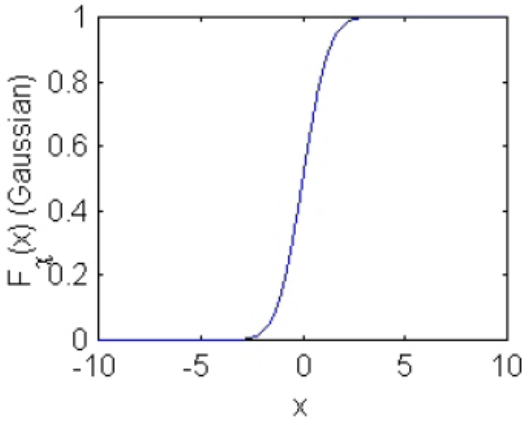
$$f_\chi(x) \geq 0 \Rightarrow F_\chi(x) \geq 0, \; F_\chi(x + dx) - F_\chi(x) \geq 0$$

$F_\chi(x)$ is a non-decreasing function.

$$\int_{-\infty}^{+\infty} f_\chi(t)dt = 1 \Rightarrow f_\chi(x)|_{x=+/-\infty} = 0$$
$$F_\chi(x)|_{x=+\infty} = 1$$
$$F_\chi(x)|_{x=-\infty} = 0$$

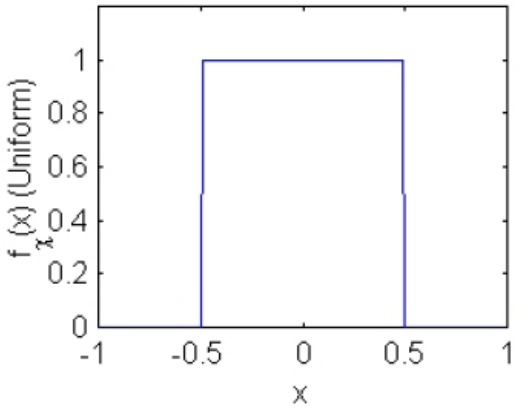
# Sürekli Genlik Rastlantı Değişkenleri

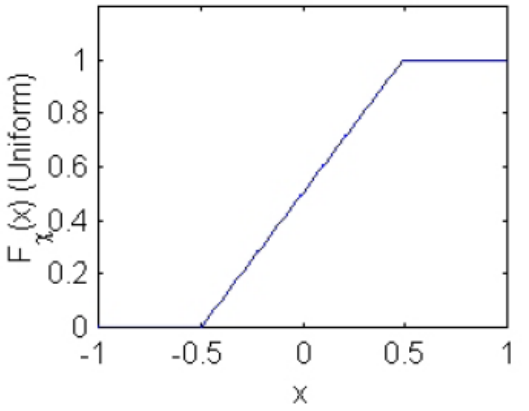



Gaussian:

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

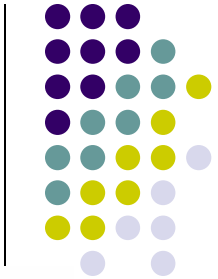



Uniform $(a < b)$:

$$f_X(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

# Ortalama ve Değişinti



- Mean $(\mu)$:

$$\mu = \int_{-\infty}^{+\infty} x f_\chi(x) dx$$

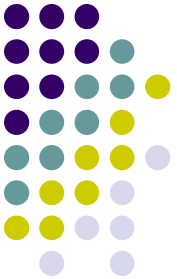
Analogy: Average price of apples

  - "I bought $f_\chi(x)dx$ many apples at a price of $x$, ..."
  - "Total price I paid: $P = \int_{-\infty}^{+\infty} x f_\chi(x)dx$.
  - "Total number of apples I purchased: $N = \int_{-\infty}^{+\infty} f_\chi(x)dx = 1$.
  - "My average price for the overall purchase: $\mu = P/N$.

- Variance $(\sigma^2)$:

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f_\chi(x) dx$$

# Ayrık Genlik Rastlantı Değişkenleri



- Let $\Theta$ be a discrete amplitude random variable.
  $\Theta = x_i$ for some $i$, $\ldots, -1, 0, 1, \ldots$.
  $x_i$ are a sequence of possible values for $\Theta$.

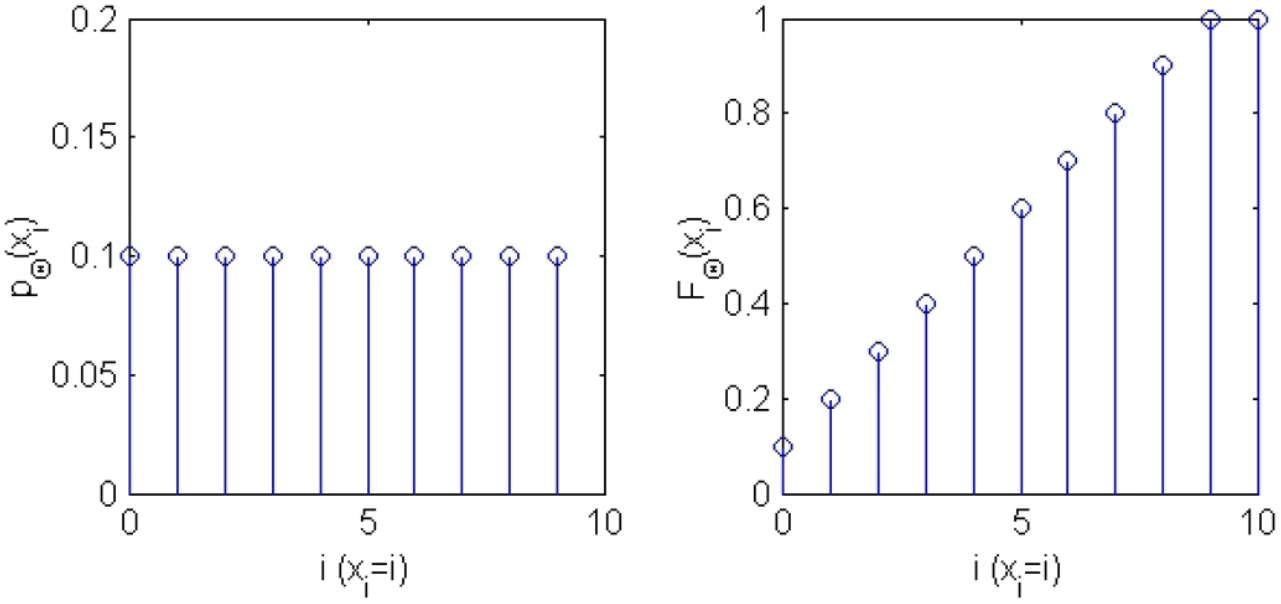
  $p_\Theta(x_i)$: the probability mass function of $\Theta$,
  $F_\Theta(x_i)$: the probability distribution function of $\Theta$.

$$p_\Theta(x_i) = \text{Probability}(\Theta = x_i)$$
$$F_\Theta(x_i) = \text{Probability}(\Theta \leq x_i)$$

- Properties:

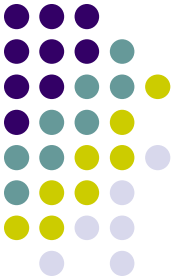
$$F_\Theta(x_i) = \sum_{j=-\infty}^{j=i} p_\Theta(x_j)$$
$$p_\Theta(x_i) = F_\Theta(x_i) - F_\Theta(x_{i-1}) \geq 0$$
$$\sum_{j=-\infty}^{j=+\infty} p_\Theta(x_j) = 1$$

# Ayrık Genlik Rastlantı Değişkenleri




The probability mass and distribution functions for a uniform, discrete amplitude random variable.

# Olasılık Yoğun Fonk. Olarak Histogram



- For a given image $\mathbf{A}$, consider the image pixels as the realizations of a discrete amplitude random variable "$A$".
  - For example suppose we toss a coin (Heads=255 and Tails=0) $N \times M$ times and record the results as an $N$ by $M$ image matrix.
- Define the sample probability mass function $p_A(l)$ as the probability of a randomly chosen pixel having the value $l$.

$$p_A(l) = \frac{h_A(l)}{NM}$$

- Note that the sample mean and variance we talked about in Lecture 2 can be calculated as:

$$m_A = \sum_{l=0}^{255} l p_A(l)$$

$$\sigma_A^2 = \sum_{l=0}^{255} (l - m_A)^2 p_A(l)$$

# Histogram Eşitleme

- Amaç: İmgedeki düşük görünürlüğü iyileştirmek.
- Olasılık dağılımına bağlı olarak doğrusal olmayan dönüşüm gerçekleştirilir.
- Bu sayede, bulunma olasılığı yüksek pikseller arası fazlaca açılırken, düşük olasılıklı seviyeler birbirine daha yakın hale gelir.

$$cdf(v) = round\left( \frac{cdf(v) - cdf_{\min}}{(M \times N) - cdf_{\min}} \times (L-1) \right)$$

# Histogram Eşitleme

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

blok

$$cdf(v) = round\left(\frac{cdf(v) - cdf_{\min}}{(M \times N) - cdf_{\min}} \times (L-1)\right)$$

$$cdf(v) = round\left(\frac{cdf(v)-1}{64-1} \times 255\right)$$

$$cdf(78) = round\left(\frac{46-1}{63} \times 255\right) = 182$$

histogram

$$cdf(154) = round\left(\frac{64-1}{63} \times 255\right) = 255$$

| Value | Count | Value | Count | Value | Count | Value | Count | Value | Count |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 64 | 2 | 72 | 1 | 85 | 2 | 113 | 1 |
| 55 | 3 | 65 | 3 | 73 | 2 | 87 | 1 | 122 | 1 |
| 58 | 2 | 66 | 2 | 75 | 1 | 88 | 1 | 126 | 1 |
| 59 | 3 | 67 | 1 | 76 | 1 | 90 | 1 | 144 | 1 |
| 60 | 1 | 68 | 5 | 77 | 1 | 94 | 1 | 154 | 1 |
| 61 | 4 | 69 | 3 | 78 | 1 | 104 | 2 | | |
| 62 | 1 | 70 | 4 | 79 | 2 | 106 | 1 | | |
| 63 | 2 | 71 | 2 | 83 | 1 | 109 | 1 | | |

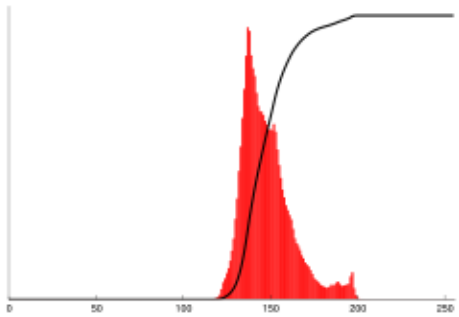
| Value | cdf | Value | cdf | Value | cdf | Value | cdf | Value | cdf |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 64 | 19 | 72 | 40 | 85 | 51 | 113 | 60 |
| 55 | 4 | 65 | 22 | 73 | 42 | 87 | 52 | 122 | 61 |
| 58 | 6 | 66 | 24 | 75 | 43 | 88 | 53 | 126 | 62 |
| 59 | 9 | 67 | 25 | 76 | 44 | 90 | 54 | 144 | 63 |
| 60 | 10 | 68 | 30 | 77 | 45 | 94 | 55 | 154 | 64 |
| 61 | 14 | 69 | 33 | 78 | 46 | 104 | 57 | | |
| 62 | 15 | 70 | 37 | 79 | 48 | 106 | 58 | | |
| 63 | 17 | 71 | 39 | 83 | 49 | 109 | 59 | | |

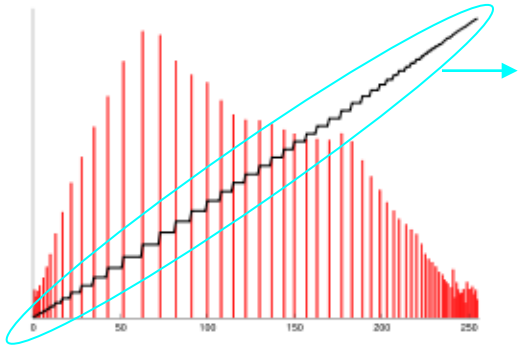
cdf

$$\begin{bmatrix} 0 & 12 & 53 & 93 & 146 & 53 & 73 & 166 \\ 65 & 32 & 12 & 215 & 235 & 202 & 130 & 158 \\ 57 & 32 & 117 & 239 & 251 & 227 & 93 & 166 \\ 65 & 20 & 154 & 243 & 255 & 231 & 146 & 130 \\ 97 & 53 & 117 & 227 & 247 & 210 & 117 & 146 \\ 190 & 85 & 36 & 146 & 178 & 117 & 20 & 170 \\ 202 & 154 & 73 & 32 & 12 & 53 & 85 & 194 \\ 206 & 190 & 130 & 117 & 85 & 174 & 182 & 219 \end{bmatrix}$$

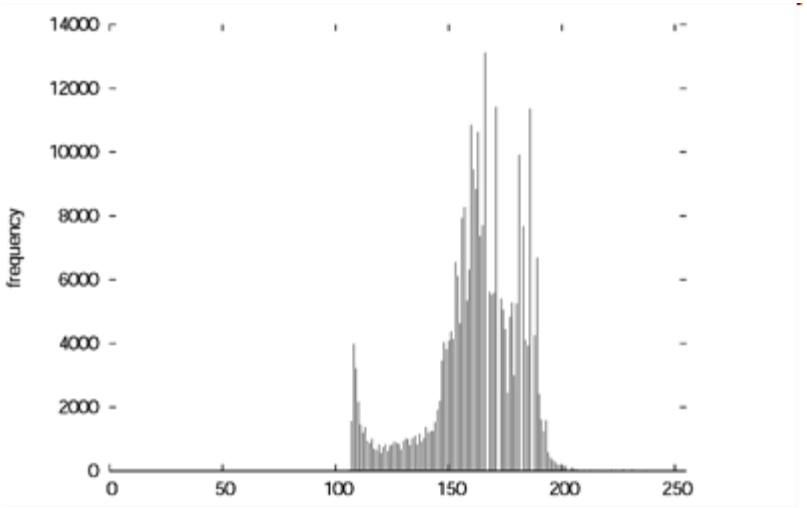
# Histogram Eşitleme

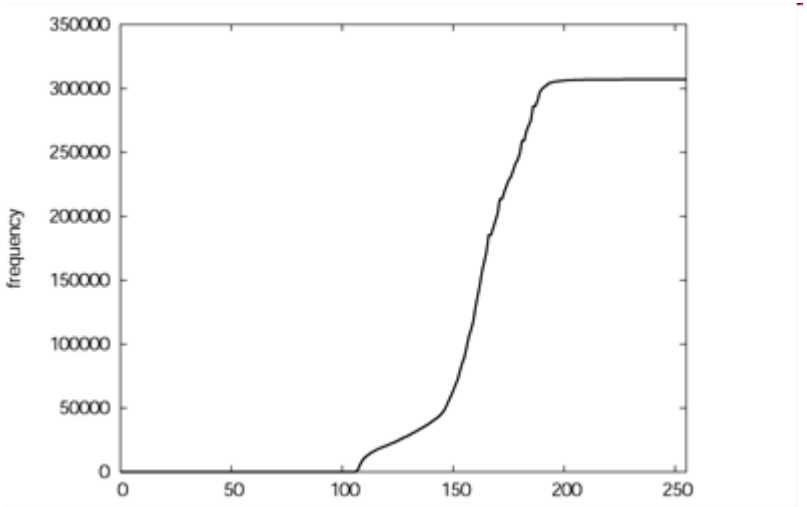
- İmgenin olasılık dağılım fonksiyonu doğrusallaştırılmaktadır.



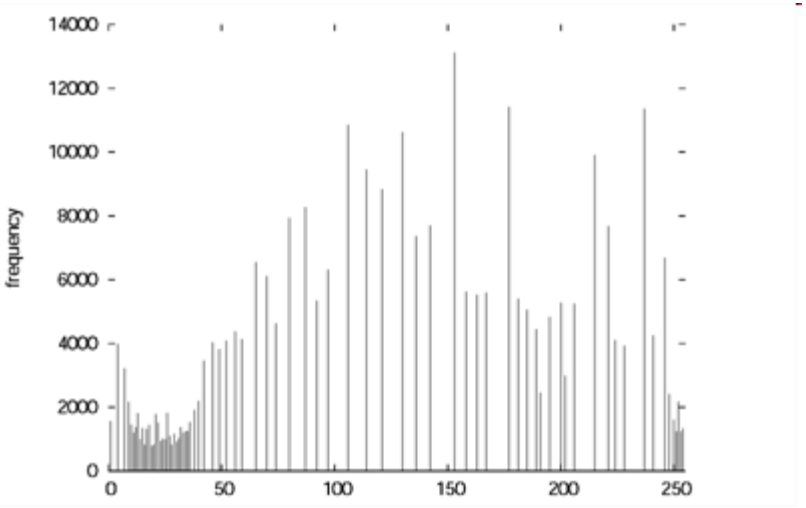



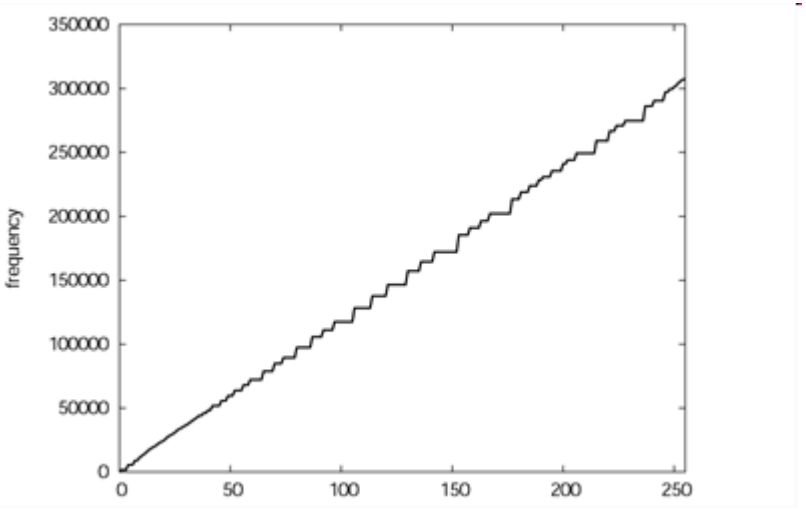
# Histogram Eşitleme



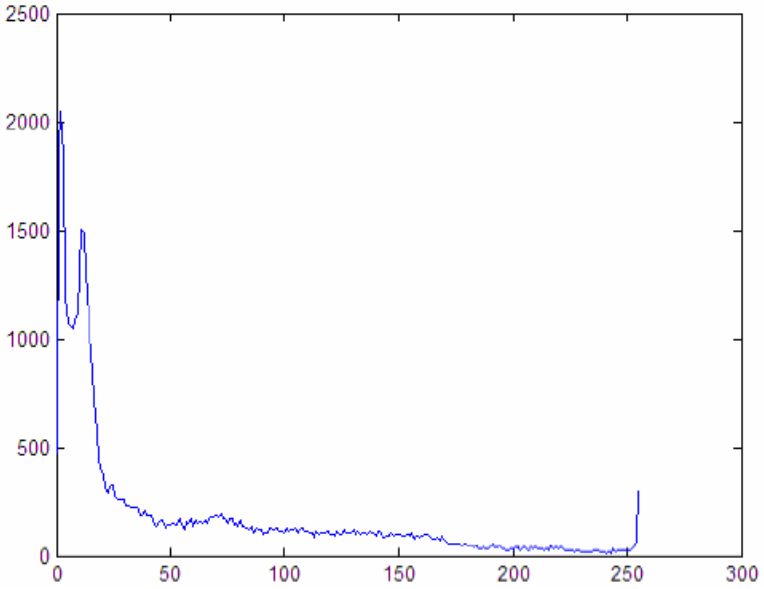



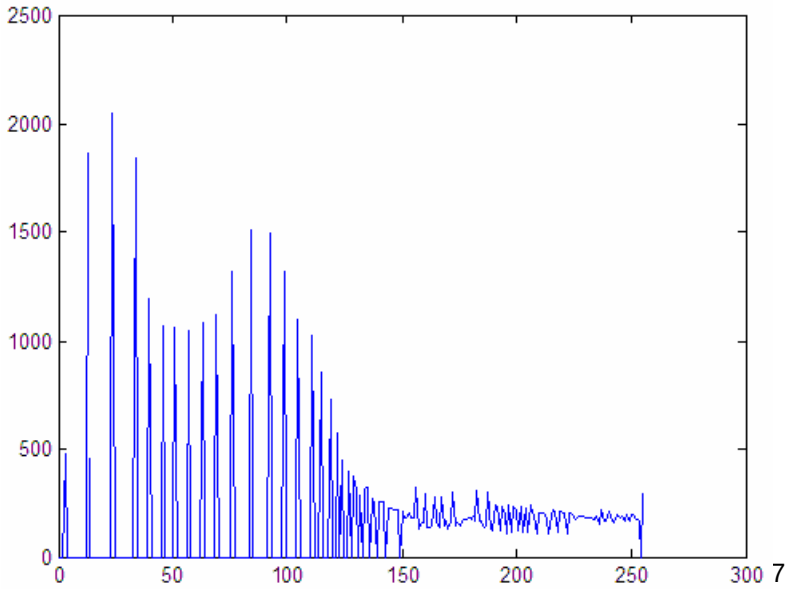
# Histogram Eşitleme



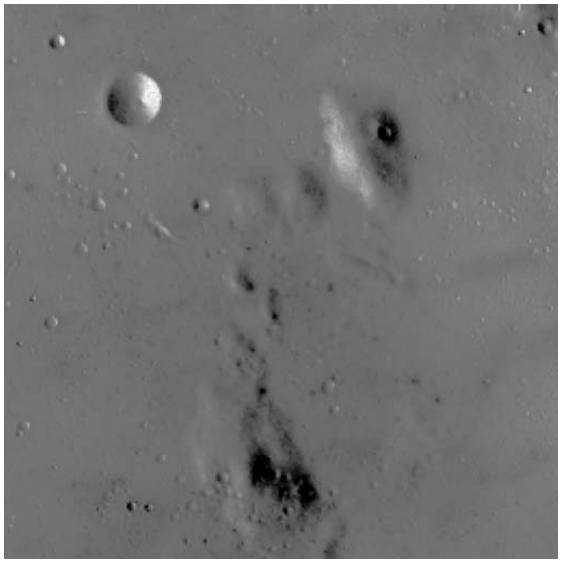



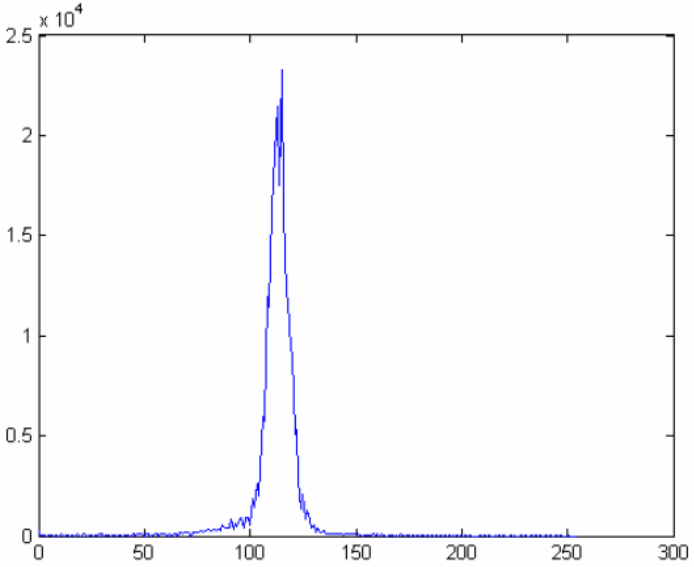
# Histogram Eşitleme

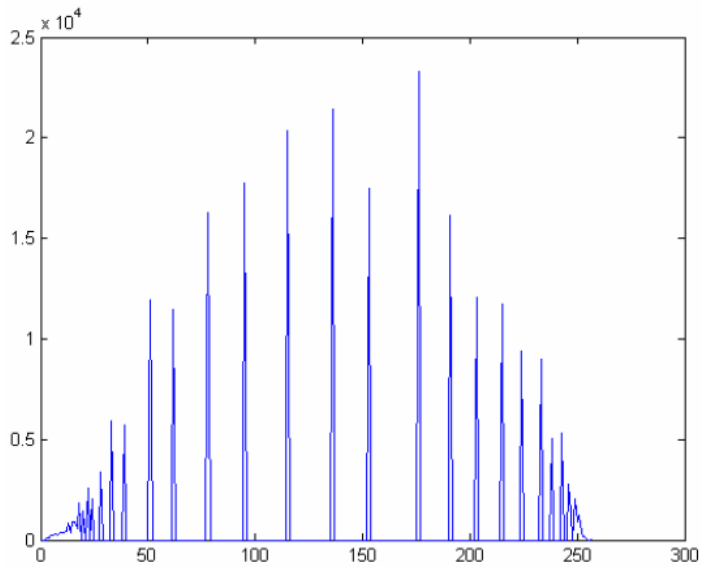


# Histogram Eşitleme

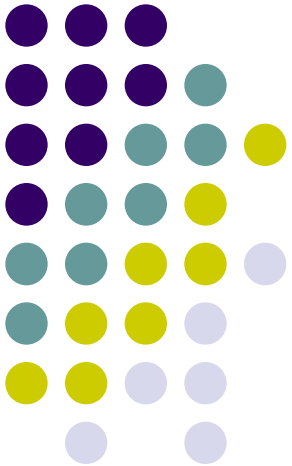




# İMGE İŞLEME
# Ders-4

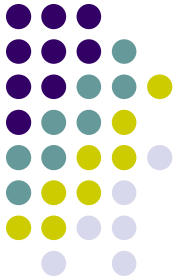
## Piksel Komşuluk İşlemleri




(Yrd. Doç. Dr. M. Kemal GÜLLÜ)

Dersin web sayfası: http://mf.kou.edu.tr/elohab/kemalg/imge_web/odev.htm

# İmgenin Ortalama ve Değişintisi



- Bir imgenin ortalaması (mean):

$$\mu = \mathrm{E}(X) = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

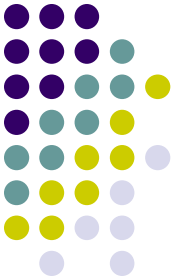
- Bir imgenin değişintisi (variance):

$$\sigma^2 = Var(X) = \mathrm{E}\left((X-\mu)^2\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

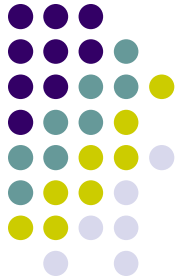
- MATLAB'da 2-boyutlu matrisin ortalamasını almak için **mean2** işlevi kullanılaktadır.
- Değişinti hesabı için **std2** standart sapma bulma işlevi kullanılmaktadır. Daha sonra standart sapmanın karesi alınarak değişinti bulunabilmektedir.

# İmgenin Ortalama ve Değişintisi



- MATLAB'da 8 bitlik bir I imgesi açıp, imgenin her bir pikseline erişerek etrafından nxn boyutlu bir blok alın ve bloğun standart sapmasını hesaplayın.
- I ile aynı boyutta oluşturacağınız I2 imgesinin ilgili pikseline bulunan standart sapma değerini yazınız.
- Bu işlemi imgedeki bütün pikseller için yapınız.

# Piksel Komşuluk İşlemleri



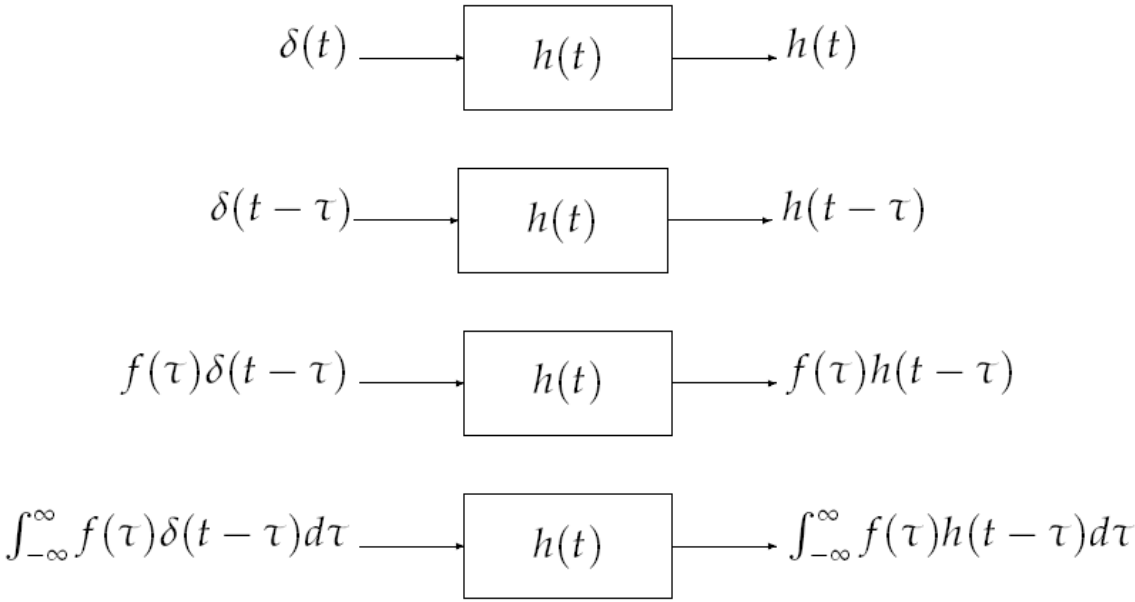
- Her bir piksel için yeni bir değer hesaplanmaktadır.
- İlgili pikselin yeni değeri, komşu piksellerin değerleri de dikkate alınarak bulunur.
- Kullanılacak piksellerin ağırlıkları, yapılacak işleme bağlı olarak değişmektedir.
- Kenar bulma, gürültü giderme, imge keskinleştirme, yumuşatma gibi işlemlerde kullanılmaktadır.
- Hesapsal yükü, nokta işlemlerine göre oldukça fazla olabilmektedir.

# Evrişim (Convolution)

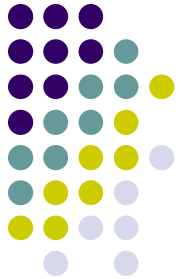
- İki fonksiyonun etkileşimi olarak ifade edilebilir.

$$f * g = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau$$

- İmge işlemede sıkça kullanılmaktadır.
- Sistemin, giriş işaretine etkisini vermektedir.



# Evrişim (Convolution)



- Evrişimin ayrık zamanlı 2-boyutlu ifadesi:

$$g(x,y) = k * f$$

$$= \sum_{i=-m}^{m} \sum_{j=-n}^{n} k(i,j)\, f(x-i, y-j)$$

$k$, evrişim çekirdeği (convolution kernel)

$f$, giriş imgesi

$g$, çıkış imgesi

$(x,y)$, ilgili piksel konumu

$(2m+1, 2n+1)$, çekirdeğin yatay ve düşey uzunluğu

- Evrişim çekirdeği (kernel) genelde , evrişim maskesi (convolution mask), evrişim penceresi (convolution window) olarak da adlandırılabilmektedir.

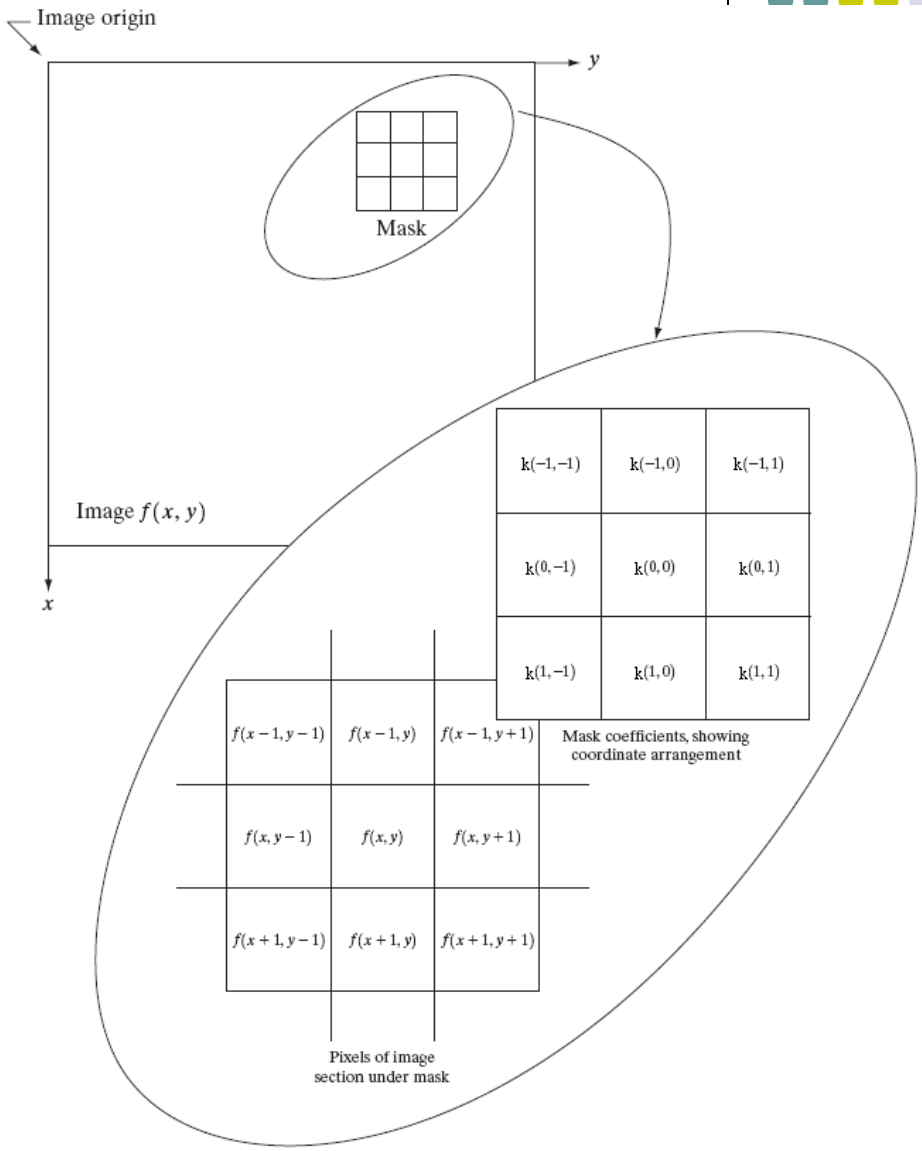
# Evrişim (Convolution)

$$g(x,y) = k * f$$

$$= \sum_{i=-m}^{m} \sum_{j=-n}^{n} k(i,j)\, f(x-i, y-j)$$

$$g(x,y) = k(-m,-n)\, f(x+m, y+n)$$
$$+k(-m+1,-n+1)\, f(x+m-1, y+n-1)$$
$$+\ldots$$
$$+k(m,n)\, f(x-m, y-n)$$

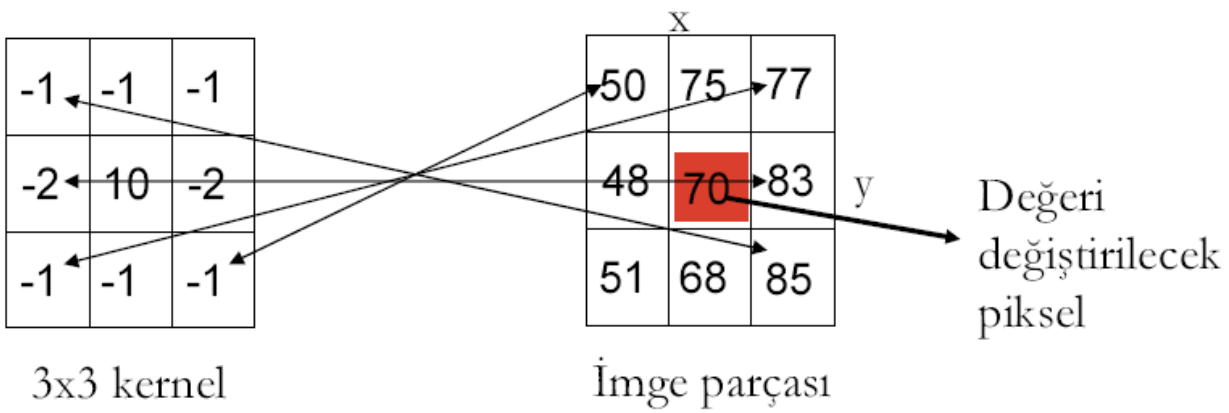
$$g(x,y) = k(-1,-1)\, f(x+1, y+1)$$
$$+k(-1,0)\, f(x+1, y)$$
$$+\ldots$$
$$+k(1,1)\, f(x-1, y-1)$$



# Evrişim (Convolution)

$$g(x,y) = k * f$$

$$= \sum_{i=-m}^{m} \sum_{j=-n}^{n} k(i,j)\, f(x-i, y-j)$$

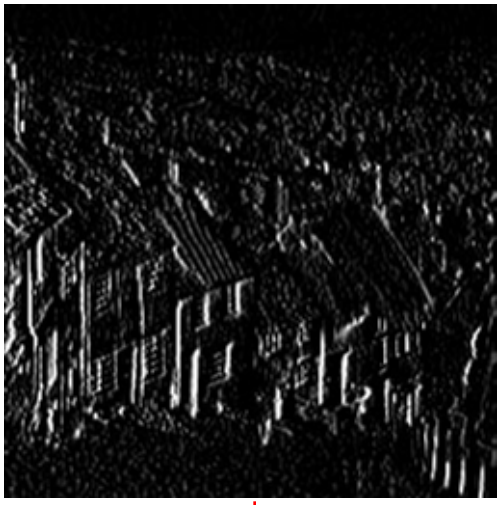



$$g(x,y)=k(-1,-1)f(x+1,y+1) + k(0,-1)f(x,y+1) + k(1,-1)f(x-1,y+1) + k(-1,0)f(x+1,y)$$
$$+ k(0,0)f(x,y) + k(1,0)f(x-1,y) + k(-1,1)f(x+1,y-1) + k(0,1)f(x,y-1)$$
$$+k\,(1,1)f(x-1,y-1)$$

$$g(x,y)=(-1x85) + (-1x68) + (-1x51) + (-2x83) + (10x70) + (-2x48) + (-1x77)$$
$$+ (-1x75) + (-1x50) = 32$$

# Evrişim (Convolution)

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



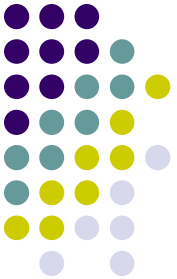
Giriş imgesi

Evrişim çekirdeği

Çıkış imgesi

- MATLAB'da 2-boyutlu evrişim **conv2** işlevi ile yapılabilmektedir.
- Bunun yanında imge süzgeçlerken genellikle **imfilter** işlevi kullanılmaktadır.

# Evrişim (Convolution)



Evrişim işleminde kenar bölgelerindeki taşma durumunda olası işlemler:

- Kenar bölgelerini işlememe,
- Kenar bölgelerini kesme,
- Kenar bölgelerinde evrişim çekirdeğini kırpma,
- Kenar bölgelerini aynen kopyalama (imge boyutları büyür),
- Kenar bölgelerini aynalayarak kopyalama (imge boyutları büyür)...

Hesapsal yük:

- $(m,n)$ boyutlu bir evrişim çekirdeği kullanıldığında bir piksel için çıkış değerinin hesaplanmasında gerekli işlem sayısı:

$$(m \times n)[çarpma] + (m \times n - 1)[toplama]$$

# Evrişim (Convolution)

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

Delta fonksiyonu

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

Kaydır ve çıkart

# Evrişim (Convolution)

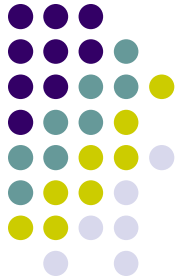
$$\begin{bmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{bmatrix}$$

Kenar bulma

$$\begin{bmatrix} -k/8 & -k/8 & -k/8 \\ -k/8 & k+1 & -k/8 \\ -k/8 & -k/8 & -k/8 \end{bmatrix}$$

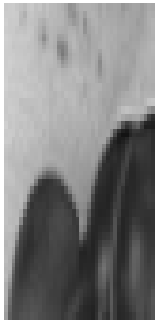
Kenar pekiştirme

# Uzamsal Frekans Kavramı



- İmgede pikseller arasındaki yumuşak geçişler *uzamsal düşük frekanslara* karşılık gelir.

- Sert geçişler (kenarlar, nesne sınırları...) *uzamsal yüksek frekanslara* karşılık gelir.

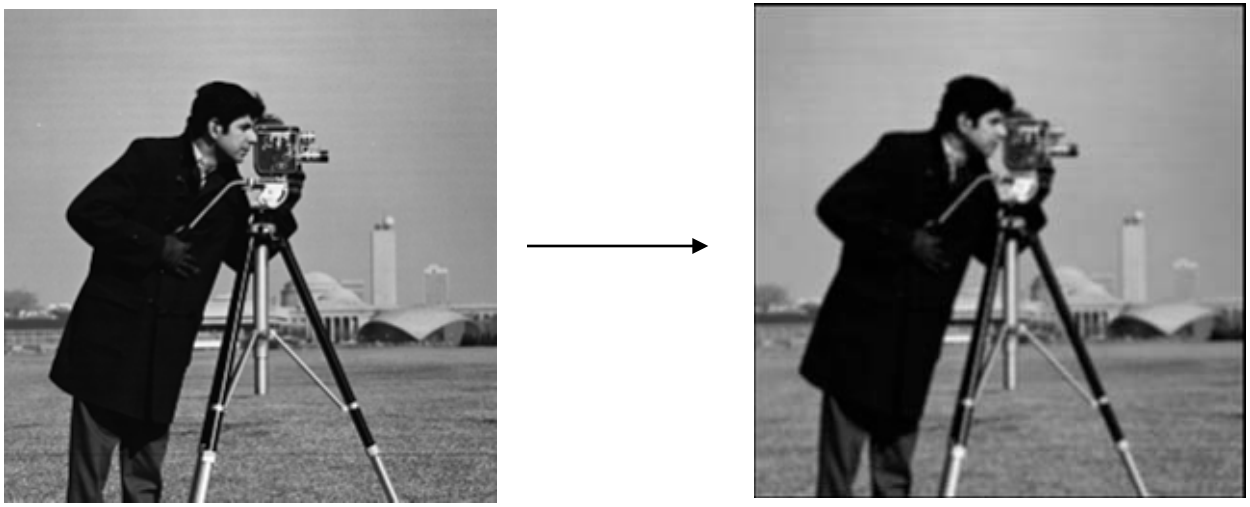


# Evrişim (Convolution)-Yumuşatma

- En temel evrişim çekirdeğidir.
- İmgedeki gürültü etkilerini azaltır.
- Kenarları yumuşatır.

$$1/9 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$1/25 \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



# Evrişim (Convolution)-Yumuşatma

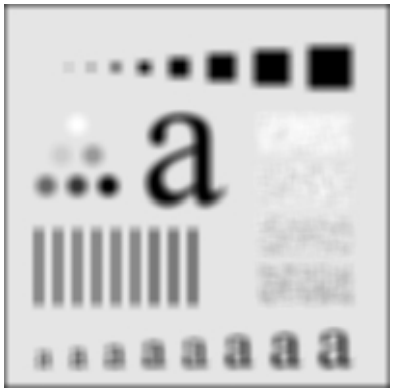
- Çekirdek boyutunun yumuşatmaya etkisi:

Orjinal imge

3x3

5x5

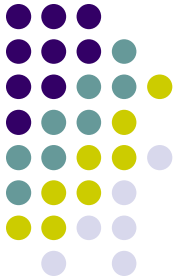
9x9



15x15

35x35

# Evrişim (Convolution)-Yumuşatma



- Ağırlıklı ortalama alma işlemi de yapılabilmektedir.

$$g(x,y)=\frac{\sum_{i=-m}^{m}\sum_{j=-n}^{n} w(i,j)\, f(x-i,y-j)}{\sum_{i=-m}^{m}\sum_{j=-n}^{n} w(i,j)} \qquad 1/15\times\begin{bmatrix}1 & 2 & 1\\ 2 & 3 & 2\\ 1 & 2 & 1\end{bmatrix}$$

- Kenar bölgelerindeki yumuşamayı azaltmak için kontrollü ortalama alma yapılabilir.

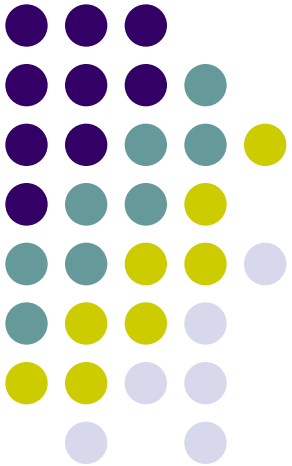
$$g(x,y)=\begin{cases}\dfrac{1}{ws\times ws}\sum_i\sum_j f(x-i,y-j) & , \left| f(x,y)-\dfrac{1}{ws\times ws}\sum_i\sum_j f(x-i,y-j)\right| < T \\ f(x,y) & , \text{değilse}\end{cases}$$

↓

MATLAB'da uygulayınız

# İMGE İŞLEME
# Ders-5

## Piksel Komşuluk İşlemleri-2




(Yrd. Doç. Dr. M. Kemal GÜLLÜ)

Dersin web sayfası: http://mf.kou.edu.tr/elohab/kemalg/imge_web/odev.htm

# Ortanca (Median) Süzgeç

- Süzgeçleme işlemi, pencere içerisindeki piksellerin sıralanması temelinde yapmaktadır.
- Doğrusal olmayan bir süzgeçlemedir.
- Dürtü ve tuz-biber gürültülerinin giderilmesinde etkin başarım sağlamaktadır.
- İmgenin kenar bölgelerini bozmaktadır.

| | | | | |
|---|---|---|---|---|
| 25 | 28 | 34 | x | x |
| 45 | 41 | 56 | x | x |
| 38 | 46 | 29 | x | x |
| x | x | x | x | x |
| x | x | x | x | x |

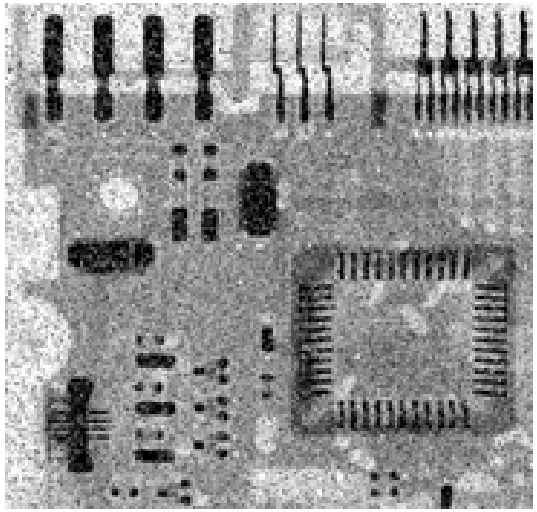
25, 28, 29, 34, 38, 41, 45, 46, 56

Yeni piksel değeri

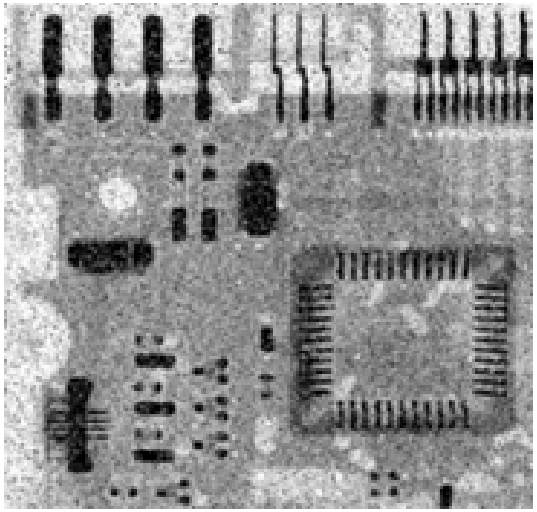
- MATLAB'da imgeye gürültü eklemek için **imnoise** işlevi kullanılmaktadır.

# Ortanca (Median) Süzgeç

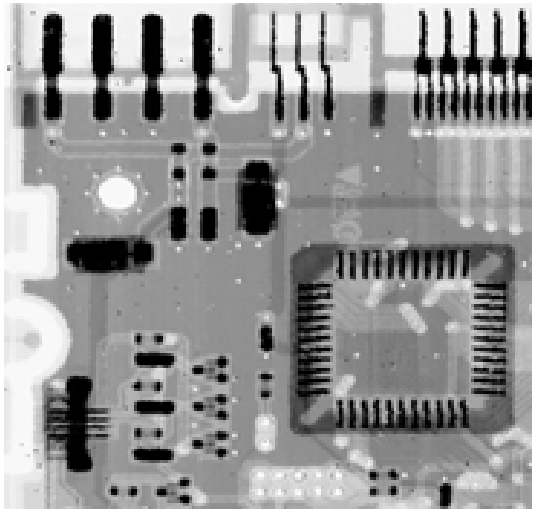
- Tuz ve biber gürültüsünün (salt and pepper noise) ortanca süzgeç ile giderilmesi



Gürültü eklenmiş imge



3x3 ortalama süzgeç ile gürültü giderme



3x3 ortanca süzgeç ile gürültü giderme

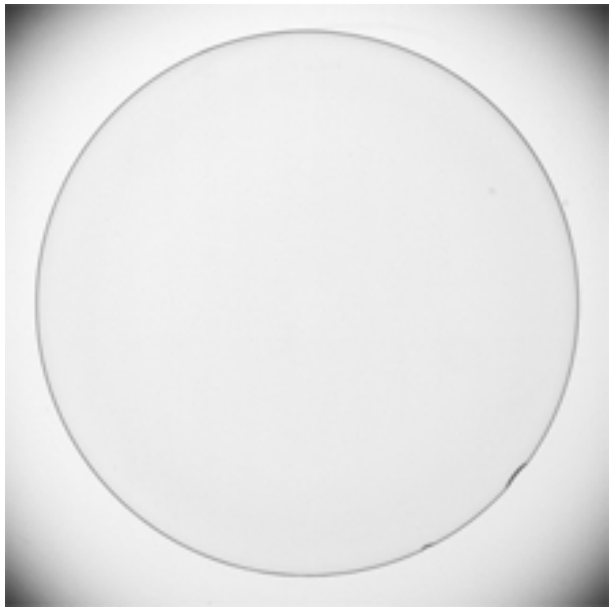
- MATLAB'da ortanca süzgeçleme için **medfilt2** işlevi kullanılmaktadır.

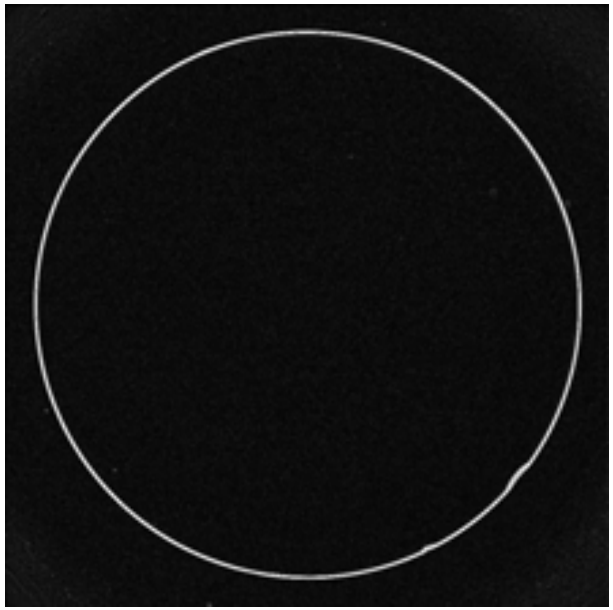
# Evrişim (Convolution)-Keskinleştirme

- Kenar: İmgedeki keskin ışıklılık değişimleridir.
- Keskinleştirme işlemindeki temel hedef detayları daha görünür hale getirmek ve bulanık bölgelerden detay çıkartmaya çalışmaktır.
- Keskinleştirme işlemi, çıktı kalitesini arttırma, tıbbi görüntüleme, endüstriyel denetim, kendi kendine dolaşan robot gibi uygulamalarda kullanılmaktadır.

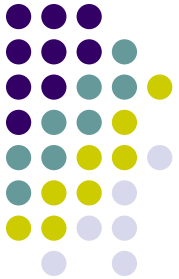
# Evrişim (Convolution)-Kenar Bulma

- Kenar bulma imge içerisindeki anlamlı kenarların bulunması olarak ifade edilmektedir.
- Bölütlemede nesne sınırlarının bulunması, tanımada örüntü çıkartma, hareket analizinde bölgeleri takip etme gibi uygulamalarda kenar bulma kullanılmaktadır.





# Evrişim (Convolution)-Keskinleştirme



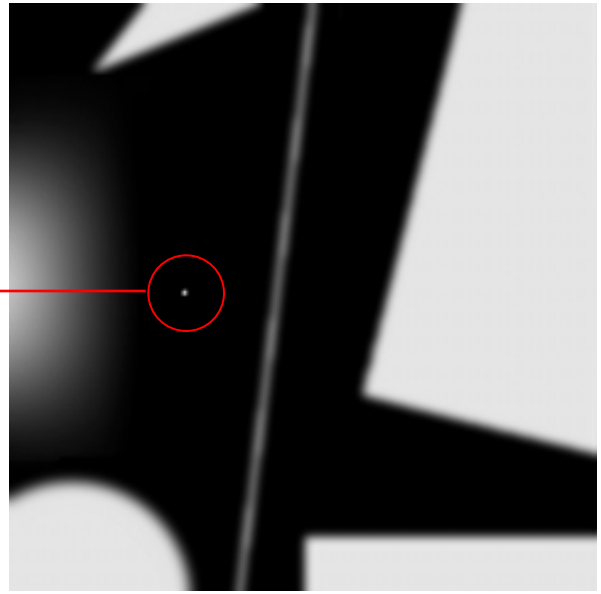
- Keskinleştirme işlemi, sayısal türevleme kullanılarak farklı yollarla yapılabilmektedir.
- Temelde, türev alma işleminin yanıtı, imge operatörün uygulandığı noktadaki süreksizlik ile orantılıdır.
- Tek boyutlu bir fonksiyon için 1. dereceden türev:

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

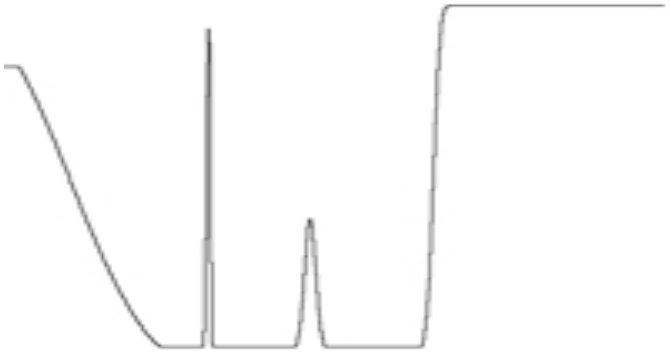
- 2. dereceden türev:

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

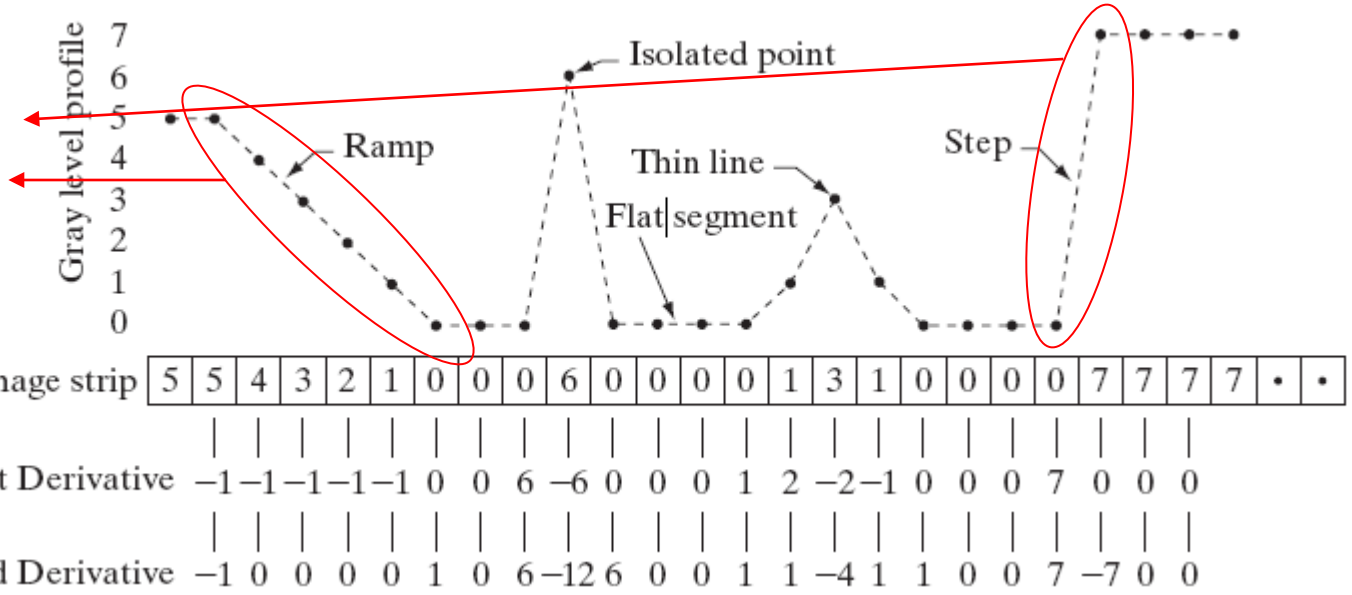
# Evrişim (Convolution)-Keskinleştirme



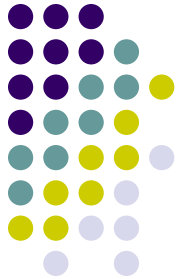
gürültü



1. ve 2. dereceden türev değerlerini karşılaştırınız



# Evrişim (Convolution)-Keskinleştirme



- 1. dereceden türev kalın kenarlar üretmektedir.
- 2. dereceden türev, detay bölgelerinde daha fazla tepki vermektedir (örn; dikey ince çizgi ve gürültü bölgeleri).
- 2. dereceden türev, kenar bölgelerinde ve nokta değişimlerinde daha fazla tepki vermesinden dolayı, keskinleştirmede daha fazla tercih edilmektedir.

2. Dereceden Türev Kullanımı - Laplacian Filtresi:

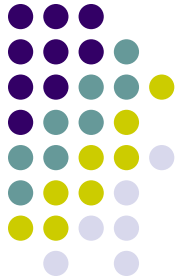
$$\partial^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

ayrık şekilde:

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

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

# Evrişim (Convolution)-Keskinleştirme



- Tek bir ifade ile 2-B Laplacian:

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

$$\downarrow$$

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

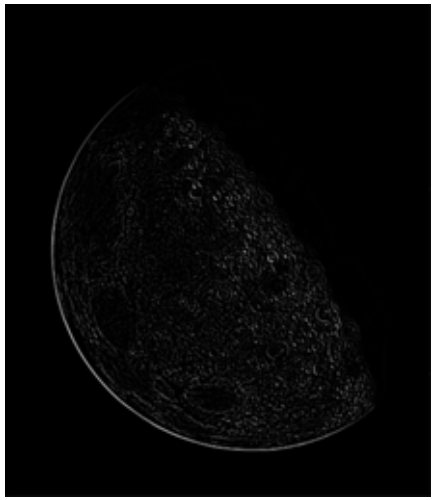
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

# Evrişim (Convolution)-Keskinleştirme

- 2-B Laplacian çekirdeği kullanılarak süzgeçlenen imge ile orjinal imge kullanılarak keskinleştirilmiş imge aşağıdaki şekilde elde edilir:

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{gerçekirdeğin merkez değeri negatif ise} \\ f(x,y) + \nabla^2 f(x,y) & \text{gerçekirdeğin merkez değeri pozitif ise} \end{cases}$$

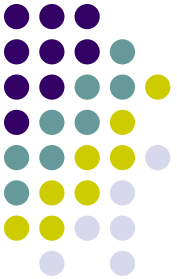
Orjinal imge



Laplacian filtrelenmiş imge

Sonuç imgesi

# Evrişim (Convolution)-Kenar Bulma



## 1. Dereceden Türev Kullanımı – Eğim (Gradyan-The Gradient):

- 2-B sütun vektörü olarak gradyan:

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Bu vektörün genliği:

$$\nabla f = \mathrm{mag}\left(\nabla \mathrm{f}\right) = \left[G_x^2 + G_y^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

- Gradyeni bulurken kare ve karekök alma işlemlerinin hesapsal yükünü azaltmak için pratikte mutlak toplam kullanılmaktadır:

$$\nabla f \approx \left|G_x\right| + \left|G_y\right|$$

# Evrişim (Convolution)-Kenar Bulma

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

$$G_x = z_8 - z_5$$
$$G_y = z_6 - z_5$$

$$\nabla f \approx |G_x| + |G_y|$$
$$\approx |z_8 - z_5| + |z_6 - z_5|$$

- Robert cross gradient operator:

$$G_x = z_9 - z_5$$
$$G_y = z_8 - z_6$$

$$\nabla f \approx |G_x| + |G_y|$$
$$\approx |z_9 - z_5| + |z_8 - z_6|$$

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

- Boyutu çift sayılardan (2x2) oluşan maske yerine 3x3 maske oluşturursak:

$$\nabla f \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

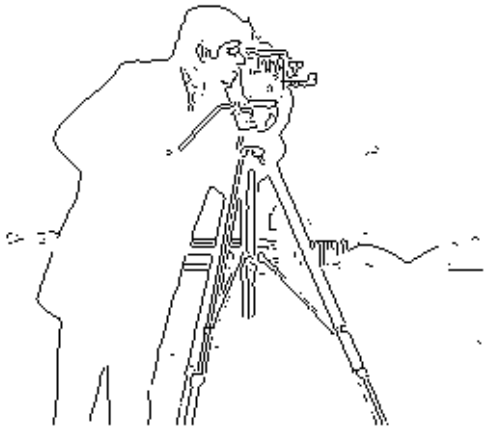
Sobel Operatörleri

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

# Evrişim (Convolution)-Kenar Bulma

- Prewitt Operatörü:

$$h_{yatay} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_{düşey} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$



- Not: Çekirdekler kullanılarak elde edilen imgenin eşiklenmesi ile ikili kenar imgesi oluşmaktadır.

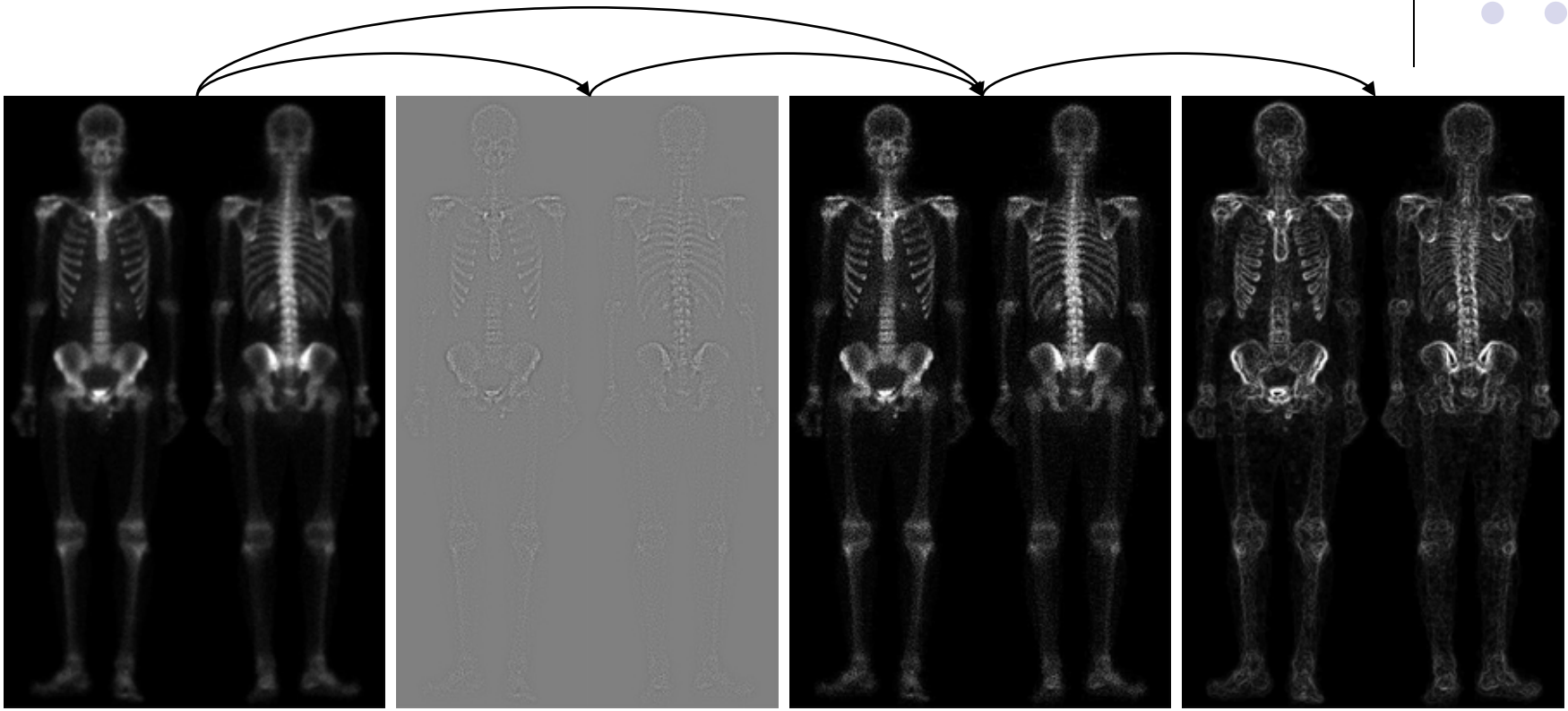
# Evrişim (Convolution)-Kenar Bulma

Kirsch Operatörü:

- Örüntü tanımada şablon eşleştirmede kullanılmaktadır.
- Kenar yönlerine çok duyarlıdır.

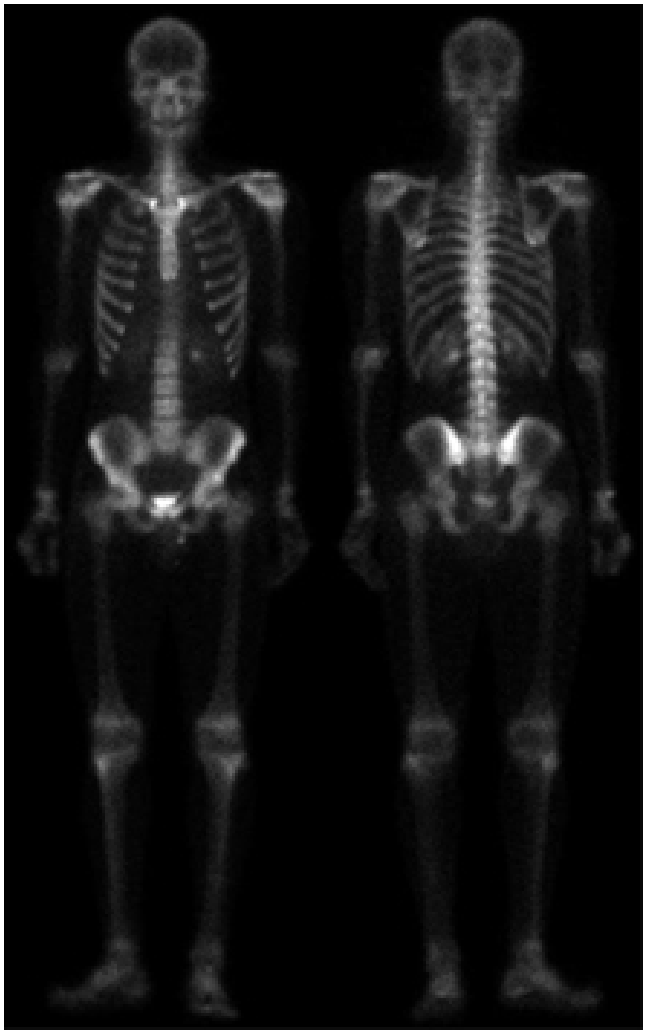
$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$$

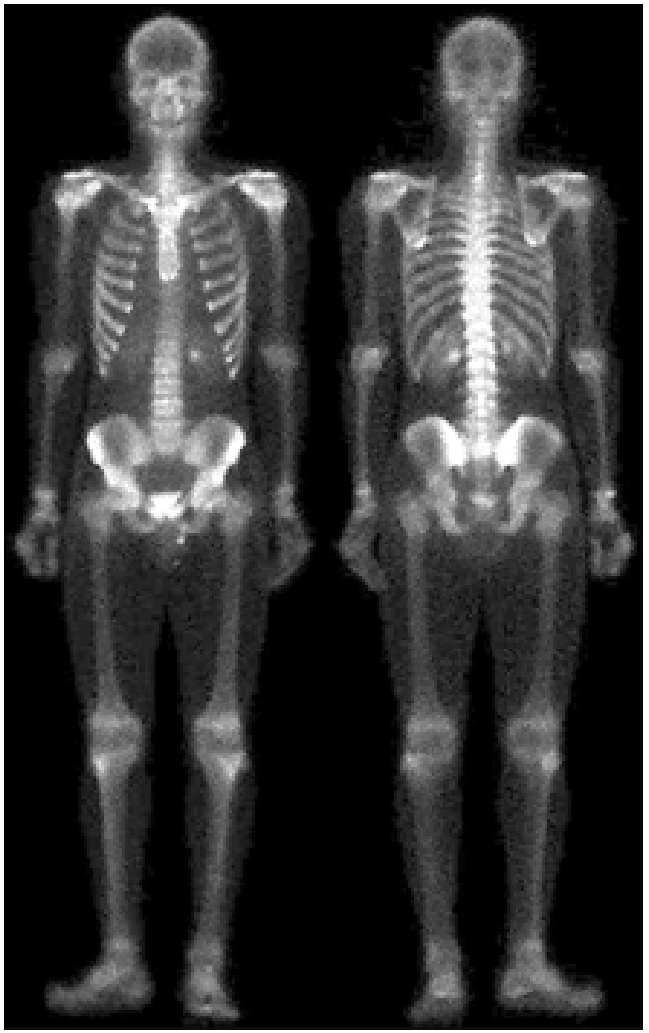
# Uygulama



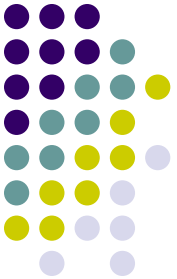
a- Orjinal imge b- Laplacian c- Orjinal + Laplacian d- Sobel

# Uygulama-devam





# İlinti (Correlation)



- İki işaret ya da imge arasındaki ilişkinin bulunması,
- Bir imgenin içerisinde imge parçası arama gibi işlemlerde kullanılmaktadır.

$$r(x,y)=\frac{\sum_{i=-m}^{m}\sum_{j=-n}^{n}\left(h(i,j)-\bar{h}\right)\left(f(x+i,y+j)-\bar{f}\right)}{\sqrt{\left(\sum_{i=-m}^{m}\sum_{j=-n}^{n}\left(h(i,j)-\bar{h}\right)^2\right)\left(\sum_{i=-m}^{m}\sum_{j=-n}^{n}\left(f(x+i,y+j)-\bar{f}\right)^2\right)}}$$

$h$ : aracak blok

$f$ : aramanın yapılacağı imge

$\bar{h}$ : blok ortalaması

$\bar{f}$ : imge ortalaması

$(x,y)$ : ilgili piksel konumu

# İlinti (Correlation)