

```

*****
% karanlık ve aydınlık yapma
x=imread('headquarters-2K.png'); %karanlık görüntü
imshow(x)

x=x-30; %daha da karanlık yapıyoruz.
figure
imshow(x)

x=x+100; %parlaklığı arttırıyoruz.
figure
imshow(x)
*****

```

İMGE TIPLERİ

1. rgb
2. gri seviye
3. binary (ikili)
4. indeksli

Görüntü işlemede kullanılan sınıflar

Name	Description
double	Double-precision, floating-point numbers in the approximate range $\pm 10^{308}$ (8 bytes per element).
single	Single-precision floating-point numbers with values in the approximate range $\pm 10^{38}$ (4 bytes per element).
uint8	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
uint16	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
uint32	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
int8	Signed 8-bit integers in the range [-128, 127] (1 byte per element).
int16	Signed 16-bit integers in the range [-32768, 32767] (2 bytes per element).
int32	Signed 32-bit integers in the range [-2147483648, 2147483647] (4 bytes per element).
char	Characters (2 bytes per element).
logical	Values are 0 or 1 (1 byte per element).

Sınıf uint8 ve mantıksal olarak görüntü işleme sürecinde yaygın olarak kullanılır ve Bunlar resim dosyasından resim okurken karşılaşılan olağan sınıflardır. Formatları-TIFF veya

JPEG gibi. Bu sınıflar her pikseli temsil etmek için 1 bayt kullanır.

Tıbbi görüntüler gibi bazı bilimsel veri kaynakları, daha dinamik olmasını gerektirir bu nedenle uint16 ya da int16 kullanılabilir.

Sınıflar arası dönüşüm

```
B=sinif_adi(A);
```

Name	Converts Input to:	Valid Input Image Data Classes
im2uint8	uint8	logical, uint8, uint16, int16, single, and double
im2uint16	uint16	logical, uint8, uint16, int16, single, and double
im2double	double	logical, uint8, uint16, int16, single, and double
im2single	single	logical, uint8, uint16, int16, single, and double
mat2gray	double in the range [0, 1]	logical, uint8, int8, uint16, int16, uint32, int32, single, and double
im2bw	logical	uint8, uint16, int16, single, and double

```
f=[-0.5 0.5;0.75 1]; %double bir matris oluşturalım.
```

```
class(f) %matrisin sınıfını öğreniyoruz.
```

```
ans =
```

```
double
```

```
>> g=im2uint8(f) %double f matrisi uint8 sınıfına çeviriyoruz.
```

```
g =
```

```
    0   128  
   191   255
```

```
>> class(g) %yeni matrisin sınıfı da uint8 olduğunu görüyoruz.
```

```
ans =
```

```
uint8
```

%Yapılan iş: Giriş datasındaki 0'dan küçük değerler 0'a set edilir. 1'den büyük değerler 255'e set edilir. Diğer giriş değerleri ise 255 ile çarpılarak dönüşüm sağlanır.

%bir başka double sınıfından h matrisini oluşturuyoruz.

```
>> h=[25 50;128 200]
```

```
h =
```

```
    25    50
   128   200
```

```
>> class(h) %Matrisin sınıfı double
```

```
ans =
```

```
double
```

%Bu işlem, giriş dizi elemanlarının herbirini 255'e bölme işlemidir. İm2double fonksiyonu; giriş datasını double sınıf dataya dönüştürür.

```
>> j=im2double(h) % double - double dönüşümü
```

```
j =
```

```
    25    50
   128   200
```

```
>> class(j)
```

```
ans =
```

```
double
```

```
>> h=uint8(h) %h matrisini uint8'e çevirelim.
```

```
h =
```

```
    25    50
   128   200
```

```
>> class(h)
```

```
ans =
```

```
uint8
```

%j=im2double(h) %uint8 matrisi double'a çevirdiğimizde 0..1 arası %değerler elde ederiz.

```
>> j=im2double(h)
```

```
j =
```

```
    0.0980    0.1961
    0.5020    0.7843
```

```
>> class(j)
```

```
ans =
```

```
double
```

```
I = mat2gray (A, [amin amax]),
```

A matrisini I görüntüsünün parlaklık değerlerine dönüştürür. Döndürülen matris I, 0.0 (siyah) ila 1.0 (tam yoğunluk veya beyaz) aralığındaki değerleri içerir.

amin ve amax, A'daki 0.0 ve 1.0'a karşılık gelen A değerleridir. Amin'den düşük değerler 0.0 olur ve amax'dan büyük değerler 1.0 olur.

```
>> A=[-2 2;0 4]
```

```
A =
```

```
    -2     2
     0     4
```

```
>> I = mat2gray(A)
```

```
I =
```

```
     0    0.6667
0.3333    1.0000
```

Binary (İkili görüntüler)

```
>> x=imread('headquarters-2.jpg');
```

```
>> bw=im2bw(x);
```

```
>> imshow(bw)
```

%belirli bir değere kadar piksel parlaklık değerlerini 0 alır, belirli bir değerden sonra 1 alır.

```
B =
```

```
      0      0.6667
0.3333      1.0000
```

```
>> A=im2bw(B,0.6)
```

```
A =
```

```
      0      1
      0      1
```

```
>> B=[1 2;3 4]
```

```
B =
```

```
      1      2
      3      4
```

```
>> A=B>2 %2 'den büyük olan değerler 1 küçükler 0 değerine atanır.
```

```
A =
```

```
      0      0
      1      1
```

%Matrislere ekrandan bakıldığında eşitliğin sağında bulunan şartı doğrulayan kümenin "1" yapıldığını görürüz.

```
a=imread('cameraman.tif');
imshow(a);
b=a<50;
figure;
imshow(b);
```

%Kendi kodlarımızla bir imgeyi ikilik görüntüye çeviriyoruz.

```
function BW_Imge = BW_ImgeYap( A,threshold )
if (max(A(:))>255)
    A=uint8(A);
end
figure,imshow(A);
title('Original image');
B=zeros(size(A,1),size(A,2));
for i=1:size(A,1)
for j=1:size(A,2)
    if(sum(A(i,j,:))>threshold)
        B(i,j)=1;
    end
end
end
end
```

```
figure(2)
imshow(B)
end
```

```
%Fonksiyonu deneyelim:
x=imread('captured.jpeg');
BW_ImgeYap(x,500)
```

Dizilerde ön değer verme (preallocation)

M bir dizi olsun.
İlk fonksiyonda başlangıç değeri verilmeden hesaplama yapılsın.

```
function y = sinfun1(M)
x=0:M-1;
for k=1:numel(x)
    y(k)=sin(x(k)/(100*pi));
end
end
```

sinfun2 'de başlangıç değerleri verilsin.

```
function y = sinfun2(M)
x=0:M-1;
y=zeros(1,numel(x));
for k=1:numel(x)
    y(k)=sin(x(k)/(100*pi));
end

end
```

```
%Yüksek değerli döngülerde başlangıç değeri verilen
fonksiyonun diğerine göre daha hızlı çalıştığı görülmektedir.
>> tic;sinfun1(500000);toc
Elapsed time is 0.065691 seconds.
>> tic;sinfun2(500000);toc
Elapsed time is 0.034077 seconds.
```

%İmge öteleme örneđi

```
x=imread('headquarters-2.jpg');
x=rgb2gray(x);
y=uint8(zeros(size(x,1),size(x,2)));
for i=1:size(x,1)
    for j=1:size(x,2)-15
        y(i,j+15)=x(i,j);
    end
end
subplot(1,2,1);
imshow(x);
title('orjinal görüntü');
subplot(1,2,2);
imshow(y);
title('ötelenmiş görüntü')
```

İmge Aritmetik Fonksiyonları

Function	Description
imadd	Adds two images; or adds a constant to an image.
imsubtract	Subtracts two images; or subtracts a constant from an image.
immultiply	Multiplies two image, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
imdivide	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
imabsdiff	Computes the absolute difference between two images.
imcomplement	Complements an image.
imlincomb	Computes a linear combination of two or more images.

```
>> x=imread('headquarters.jpg');
>> y=imread('headquarters-Defect.jpg');
>> subplot(2,3,1);
>> imshow(x)
>> subplot(2,3,2)
>> imshow(y)
>> z=imsubtract(x,y);
>> subplot(2,3,3)
>> imshow(z)
>> v=imadd(x,y);
>> subplot(2,3,4)
```

```

>> imshow(v);
>> m=immultiply(x,y);
>> subplot(2,3,5)
>> imshow(m);
>> b=imdivide(x,y);
>> subplot(2,3,6)
>> imshow(b);

absdiff=imabsdiff(x,y);
>> figure
>> imshow(absdiff);

```

% aralarındaki ilişki belirtilir.

<pre> >> complement=imcomplement(x); >> figure >> imshow(complement); >> complement(94,104,:) ans(:,:,1) = 196 ans(:,:,2) = 205 ans(:,:,3) = 204 </pre>	<pre> >> x(94,104,:) ans(:,:,1) = 59 ans(:,:,2) = 50 ans(:,:,3) = 51 </pre> <div data-bbox="1005 891 1209 972" style="border: 1px solid black; padding: 2px; width: fit-content; margin-left: auto; margin-right: auto;"> Toplam 255 </div>
---	---