



Turgut Özal Üniversitesi

Mühendislik Fakültesi

**Elektrik-Elektronik Mühendisliği
Bölümü**

EEM 403 MÜHENDİSLİK TASARIMI-I ARA RAPORU

Proje Başlığı:

Şekilbilimsel İmge İşleme Uygulamaları

Projeyi Hazırlayan:

Öğrenci No Adı Soyadı: Aziz Furkan DAĞLI

Danışman: Prof.Dr.İsmail AVCIBAŞ

19 Ocak 2015

İÇİNDEKİLER

1.	Projenin Özeti.....	3
2.	Projenin amacı ve problem Tanımı	3
3.	Literatür Özeti	3
4.	Tasarım.....	3
5.	Bu Tarihe Kadar Gerçekleştirilen Faaliyetler, İş Paketleri.....	4
6.	Projenin Biten Kısımları Ve Gelecek Çalışmalar	35
7.	Kaynakça	36

1. Projenin Özeti

İmge analizi ile bir imgeden ilgilendiğimiz bilgiyi elde edebiliriz. Özellikle bilgisayarla görü uygulamalarında ilgilenilen bilgi şekilbilimsel (morfolojik) imge işleme yöntemleri ile elde edilebilir. Bu projede şekilbilimsel imge işleme yöntemleri ile bilgisayarlı görü uygulamaları gerçekleştirilecektir. Proje 2 dönemlidir 1.dönemde imge işlemenin temelleri ve uygulamaları öğrenilecektir 2.dönemde ise belirlenen bir sorunu öğrendiğimiz bilgiler ile çözeceğimiz gerçek bir uygulama yapılacaktır.

2. Projenin amacı ve problem Tanımı

Belirlenen bilgisayarlı görü problemini bileşenlerine ayırma, formüle etme ve çözme becerisini geliştirme ve gerçek hayattaki bir sorunu imge işleme ile çözme.

3. Literatür Özeti

[1]Digital Image Processing (3rd Edition) August 31, 2007

[2]<http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/>

[3]<http://www.mathworks.com/help/matlab/ref/median.html?searchHighlight=median>

[4]<http://bilgisayarkavramlari.sadievrenseker.com/2007/11/26/ortanca-filtresi-median-filter/>

4. Tasarım

Birinci Dönem

1. İmge işlemenin temelleri ve MATLAB kullanımı
2. Şekil bilimsel imge işleme
3. Bilgisayarlı görü probleminin belirlenmesi

İkinci Dönem

5. Seçilen problemin şekilbilimsel yöntemlerle çözümünün gerçekleştirilmesi

I.Dönem

sayısal imge işlemenin öğrenilmesi İsmail AVCIBAŞ hocamın teğmin ettiği online ve yazılı kaynaklar ile sayısal imge işlemenin temelleri öğrenilecek.

Ardından Morfolojik(Şekil bilimsel yöntemler) ile ilgili kaynaklar öğrenilecek.

Kaynaklar online ve yazılı dökümanlar temin edilmiş olup çalışmalar başlatılmıştır.

II.Dönem

I.dönemde öğrendiğimiz bu bilgilerin bir sorunun çözümünde kullanılması ve gerçek hayattaki bir problemin çözülmesi.

5. Bu Tarihe Kadar Gerçekleştirilen Faaliyetler, İş Paketleri

İlk olarak Matlab'daki image processing tooları incelendi ve internetteki image processing ile ilgili örnekler araştırıldı ve matlab'da çalıştırıldı.

Sonrasında Digital Image Processing by Rafael C. Gonzalez kitabı teğmin edildi ve 1.bölüm okundu.

İsmail Hocamın belirlemiş olduğu 4 adet pdf dokümanın içerikleri. Copyright 2004 by Marcel Dekker, Inc. All Rights Reserved.

1) Fundamentals of Image Processing: MATLAB Image Processing Toolbox

2) Image Processing:Filters, Transformations,and Registration

3) Image Bölütleme

4)

Morphological Transforms And Techniques

1.pdf temel imge işleme işlemlerini içermekte ayrıca matlab da bunların nasıl yapılacağı ile ilgili örneklerden oluşur.

2.pdf Filtreleme işlemi ve matlab daki filtrelerin nasıl çalıştığını gösterir ayrıca fourier dönüşümü ile resimde yapılan işlemlerde mevcuttur.

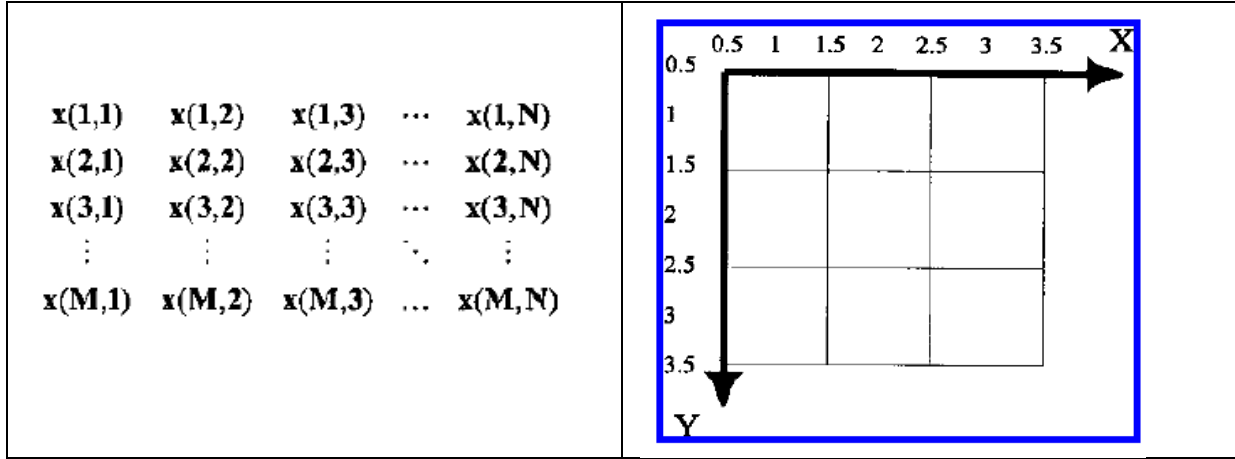
3.pdf resim bölütleme bir resimden anlamlı kısımlarını ayırma ile ilgili.

4.pdf morfoloji ile ilgili olup sadece giriş kısmı 1.dönem hedefleri arasındadır.

Resim işleme Nedir ?

Resim(imge) aslında 2 boyutlu bir matristir. x ve y kordinatları vardır.Resim(imge) işlemede bu matris üzerinden işlemler yapmamız demektir.Bu işlemler basit aritmetik işlemlerdir. [1]

Eğer bir resim 40x40 ise bu 40 satır 40 sütundan oluşan bir matris demektir.Bu matrisin her bir elemanının değerlerini görmek ister isek $I(x,y)$ ile o elemanın değeri görülebilir. x ve y sayıları tam sayılar olmak zorundadır.



Resmin koordinat sistemi ile matlab'daki matris arasında bazı farklılıklar vardır.[2]

MATLAB tarafından kullanılan dört farklı veri sınıfları, ya da kodlama şemaları vardır. Bu veri sınıfları renk, gri-tonlama, siyah ve beyaz 'ı temsil etmek için kullanılır. Bu sınıflar bellek verimi için önemlidir.

Dört farklı görüntü sınıfları veya kodlama düzenleri:

1-indexed images,

2-RGB images

3-intensity images

4-binary images.

1. ve 2. sınıflar resmi depolamada kullanılır.

3. ve 4. sınıflar resim işlemede kullanılır.

1-)Indexed Images(İndeksli Görüntü): İndeksli görüntüler bir renk haritası ve bir görüntü matrisinden oluşur. Görüntü matrisindeki sayılar görüntünün o noktalar için sahip olduğu renk değerlerine sahip renk haritasındaki satır numarasını belirtir. Renk haritasında değişik renk sayısı kadar satır ve 3 farklı renk değerine işaret eden 3 sütun vardır. Görüntü tipleri arasında dönüşümler yapılırken istenilen renk haritası uzunluğu belirlenebilir. Böylece hazırlanan görüntünün renk kalitesi istenilen şekilde ayarlanabilir. Aksi durumda MATLAB programı renk haritası uzunluğunu otomatik olarak ayarlar.

2.)Intensity Images(Yoğunluk Görüntüsü): Yoğunluk görüntüleri tek matrisden oluşur. Bu görüntü tipi adından da anlaşıldığı gibi görüntüdeki renk yoğunluğunu yani grilik derecesini belirten değerlerle ifade edilir. Eğer matris double sınıfı ise [0,1] aralığında, eğer uint8 sınıfı ise [0,255] aralığında değerlere sahiptir. 0 değeri siyah, 255 veya 1 değeri beyazı simgelemektedir.

3-)Binary Images(İkili Görüntü): İkili sistem görüntü tipinde her piksel sadece 2 değer alabilir. Yani görüntü matrisi 1 ve 0'lerden oluşur. Bu sadece beyaz ve siyah renklere sahip bir yoğunluk görüntüsü olarak kabul edilebilir.

4-)RGB Images : RGB görüntülerde de her piksel değeri için kırmızı, yeşil, mavi değerleri saklı

tutulur. Aralarındaki fark bu değerlerin renk haritasında değil farklı matrislerde saklanmasıdır. $m \times n$ 'lik görüntü için $m \times n \times 3$ ' lük bir matris tanımlanır. $m \times n$ 'lik matrislerin birinde kırmızı, birinde yeşil, birinde mavi değerleri saklı tutulur.

Görüntü üzerinde herhangi bir işlem yapılacaksa bu üç matris okunarak gerekli değişiklikler yapılır, tekrar matrisler birleştirilerek yeni görüntü elde edilir.

Matlab Komutları Ve Örnekleri

`cat(3, R, G, B)` komutu ile kırmızı, yeşil, mavi renk değerlerini taşıyan matrisler $m \times n \times 3$ 'lük görüntü matrisi haline getirilir.

cat Komutu : Matrisleri istediğimiz kombinasyonda birleştirmemizi sağlar.

Syntax

```
C = cat(dim, A, B)
C = cat(dim, A1, A2, A3, A4, ...)
```

Examples

Given

A =
1 2
3 4

B =
5 6
7 8

concatenating along different dimensions produces

1	2
3	4
5	6
7	8

C = cat(1,A,B)

1	2	5	6
3	4	7	8

C = cat(2,A,B)

5	6
7	8

1	2
3	4

C = cat(3,A,B)

Veri Dönüşümü: Hafızayı daha verimli kullanmak için ve resimleri istediğimiz formata dönüştürmek için kullanırız. Çünkü resim islemede kullanılan fonksiyonların input değerleri gri resimlerdir.

```
clear
for i=1:255
    for j=1:255
        I(i,j)=rand;
    end
end
I_uint8 = im2uint8(I); % Convert to uint8 format
I_uint16 = im2uint16(I); % Convert to uint16 format
I_double = im2double(I); % Convert to double format
```

I <255x255 double>				I_double <255x255 double>				I_uint16 <255x255 uint16>			
	1	2	3		1	2	3		1	2	3
1	0.2053	0.6692	0.8240	1	0.2053	0.6692	0.8240	1	13453	43856	53999
2	0.9947	0.8218	0.5831	2	0.9947	0.8218	0.5831	2	65185	53856	38216
3	0.3202	0.2463	0.2132	3	0.3202	0.2463	0.2132	3	20985	16141	13975
4	0.8668	0.9948	0.2619	4	0.8668	0.9948	0.2619	4	56803	65195	17166
5	0.8675	0.9647	0.7247	5	0.8675	0.9647	0.7247	5	56849	63222	47490

gray2ind

Griyi Resmi indisli resme dönüştürür.

```
I = imread('cameraman.tif');
[X, map] = gray2ind(I, 16);
imshow(X, map);
```

Matlab helpindeki class support kısmı çok önemli bu kısma bakarak argümanlar input olarak fonksiyona girilir.

rgb2ind

RGB resmi indexli resme dönüştürür.

```
[IND, map] = rgb2ind(RGB, 32);
figure('Name', 'Indexed image with 32 Colors')
imagesc(IND)
colormap(map)
axis image
zoom(4)
```

grayslice

Girdiğimiz eşik değere göre resmi griyi seviyeye çevirir.

```
I = imread('snowflakes.png');
X = grayslice(I, 16);
imshow(I)
figure, imshow(X, jet(16))
```

ind2gray

İndisli resmi gri resme çevirir.

```
load trees
I = ind2gray(X, map);
imshow(X, map)
figure, imshow(I)
```

im2bw

Resim'i eşik değerine göre siyah beyaz hale dönüştürür.

```
load trees
BW = im2bw(X,map,0.4);
imshow(X,map), figure, imshow(BW)
```

Resim Depolama ve Geri alma

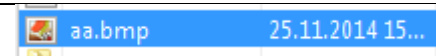
Görüntü imwrite komutu ile matlab'da oluşturduğumuz bir görüntüyü diskimize kaydedebiliriz. Aynı zamanda imread komutu ile de hafızadaki resmi matlab'a aktarabiliriz.

imwrite(matlab'da oluşturduğumuz bir görüntüyü diskimize kaydetme)

```
imwrite(X,map,'your_hdf_file.hdf','Compression','none',...
'WriteMode','append')
```

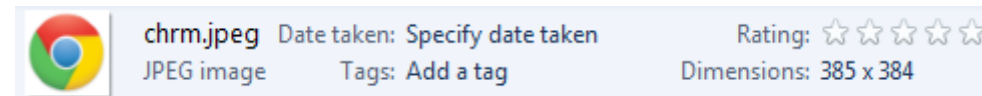
```
B=0;
for i=1:256
    for j=1:256
        dist = ((i-128)^2 + (j-128)^2)*(0.5)
        if(dist<80)
            B(i,j)=255;
        else
            B(i,j)=0;
        end
    end
end

imwrite(B,'aa.bmp')
```



imread(hafızadaki resmi matlab'a aktara)

```
[X,map] = imread('your_image.tif',6); %kullanım
```



Basic Arithmetic Operations

Burdaki kullandığımız toollar matlab'ın image processing tooları olduğu için image'e göre işlem yaparlar yani bir matrisin tümleyeni onu 255 e tamamlayan sayıdır. İki matris topladığımızda $A=[255]$ $B=[255]$ bu iki matrisi topladığımızda sonuc 500 değil 255 dir çünkü Resmin matris değerleri 0 ile 255 arasında değerler alır ve işlemler bu kurala göre çalışır.

imabsdiff :

2 Matrisin farkını alır.

```
X = uint8([ 255 10 75; 44 225 100]);  
Y = uint8([ 50 50 50; 50 50 50 ]);  
Z = imabsdiff(X,Y)
```

%Çıktı

Z =

```
    205     40     25  
     6    175     50
```

%Burda görüldüğü gibi $10-50= -40$ olmalı ama işlemlerimiz image tabanlı olduğu için matlab bunu otomatik olarak 40 a eşitledi.

Resim ile yaptığımız örnek:

```
clear  
for i=1:256  
    for j =1:256  
        I_1(i,j) = 100;  
    end  
end  
I = imread('cameraman.tif');  
figure  
imshow(I)  
  
I_1 = uint8(I_1);  
Z = imabsdiff(I,I_1);  
figure  
imshow(Z)
```



imcomplement :

Matrisin 255 e tümleyenini alır.

```
X = uint8([ 255 10 75; 44 225 100]);  
X2 = imcomplement(X)
```

```
X2 =  
      0      245      180  
     211       30      155
```

Resim ile Örnek

```
I = imread('glass.png');  
J = imcomplement(I);  
imshow(I), figure, imshow(J)
```



imadd :

iki matrsi toplar

```
X = uint8([ 255 0 75; 44 225 100]);  
Y = uint8([ 50 50 50; 50 50 50 ]);  
Z = imadd(X,Y)
```

```
Z =  
     255      50     125  
      94     255     150
```

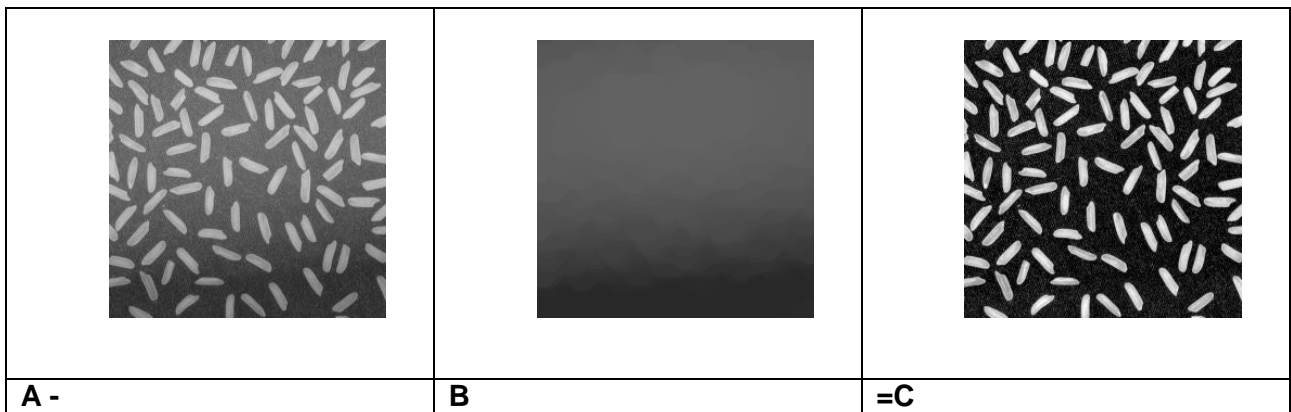


imsubtract : iki matrisi çıkartır.

```
X = uint8([ 255 10 75; 44 225 100]);
Y = uint8([ 50 50 50; 50 50 50 ]);
Z = imsubtract(X,Y)
```

```
%Çıktı1
Z =
```

```
205    0   25
   0  175   50
```

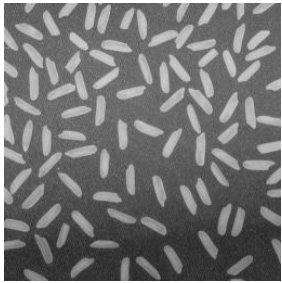

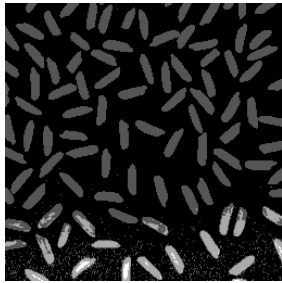


imdivide : iki matrisi böler.

```
X = uint8([ 255 10 75; 44 225 100]);
Y = uint8([ 50 20 50; 50 50 50 ]);
Z = imdivide(X,Y)
```

```
Z =
```

```
5      1      2
1      5      2
```

		
A	/B	=C

immultiply : iki matrisi çarpar.

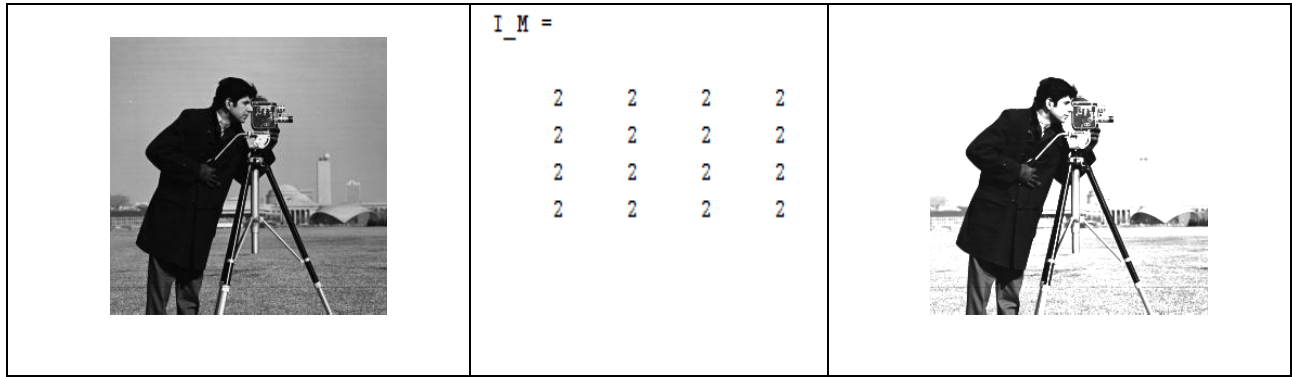
```
I = imread('moon.tif');
I16 = uint16(I);
J = immultiply(I16,I16);
imshow(I), figure, imshow(J)
```

	
J	I

imlincomb :

```
I = imread('cameraman.tif');
J = imlincomb(2,I);
imshow(J)
```

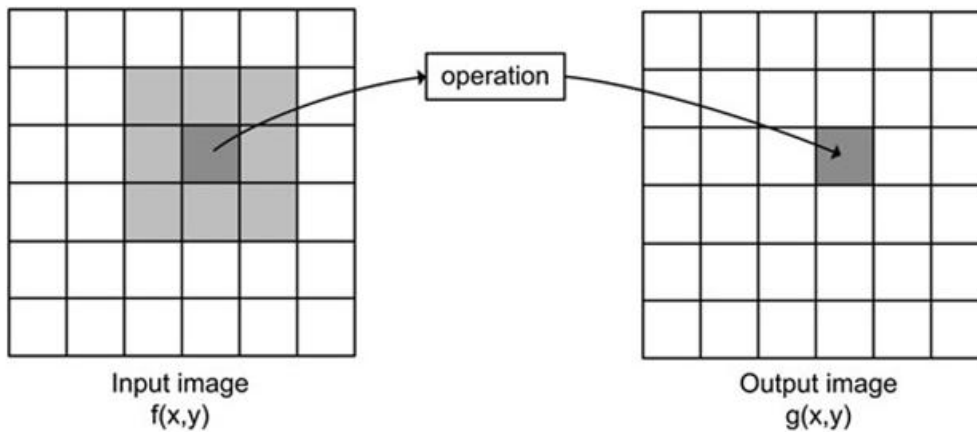
$J = 2 * I$



$Z = \text{imlincomb}(K1,A1,K2,A2,...,Kn,An)$ computes
 $K1*A1 + K2*A2 + ... + Kn*An$

Sliding Neighborhood Operations

İşlem yapılacak resim belirleyeceğimiz bir boyutta sırası ile bütün resmi tarayıp



resimde de görüldüğü gibi f matrisi 3x3 lük bir matris ile neighborhood operations yapılmış ve çıktı buna göre belirlenir.[2]

$$\text{Mean value} = \frac{205 + 204 + 204 + 206 + 0 + 208 + 201 + 199 + 205}{9} = 181.3 \simeq 181 \quad (5.1)$$

şimdi bu işlemin matlab'da nasıl yapıldığını inceleyelim.

nlfilter:

General sliding-neighborhood operations

Syntax

```
B = nlfilter(A, [m n], fun)
B = nlfilter(A, 'indexed', ...)
```

[m n]: kaçca kaçlık matrisler ile bu işlemi yapacağımız.

fun : hangi fonksiyonu uygulayacağımız mean,media,vs.... [3]

```
A = imread('cameraman.tif');
A = im2double(A);
fun = @(x) median(x(:));
B = nlfilter(A,[3 3],fun);
imshow(A), figure, imshow(B)
```

Orijinal



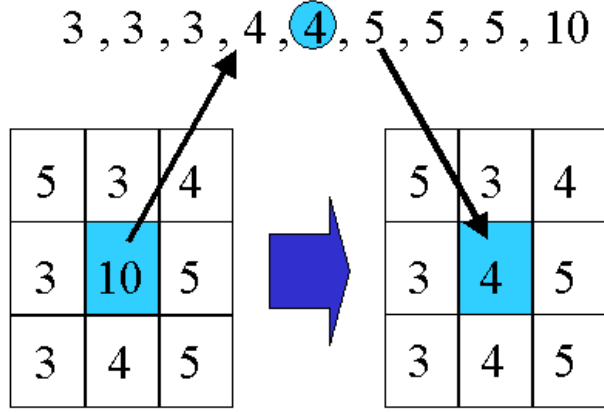
Çıktı



çıktıdaki resim orijinal resimden daha bulanık eğer m n değerleri daha büyük olsaydı netlik daha çok bozulacaktı.

Median Nedir ?

Görüntü ve sinyal işleme konularında, gürültü temizlemek için kullanılan yöntemlerden birisidir. Amaç belirli bir pencere aralığındaki sayıların ortancasını (median) alarak aşırı büyük atlamaları kaldırmaktır. Yani filtre uygulandıktan sonra resimde bulunan ve komularından belirgin şekilde ayrılan imgeciklerin (piksel) tespit edilerek temizlenmesi sağlanır.[4]



$g = [2\ 3\ 2\ 2\ 1\ 2\ 8\ 9]$

$\check{c}[0] = [2\ 2\ 3\ 2] = [2\ 2\ 3\ 2] \Rightarrow 2$ // burada ilk sayıyı tekrar ettik çünkü katar 3 olmalı ve şayet ilk sayıyı 2 kere almazsak ilk sayı için 3 adet sayımız olmayacaktır.

$\check{c}[1] = [2\ 3\ 2\ 2] = [2\ 2\ 3\ 2] \Rightarrow 2$ // ilk dizi giriş dizisinin ilk 3 sayısıdır (katar 3 olduğu için) ikinci dizi ise sıralanmış halidir. ve sonuç olarak ortanca değer 2 bulunur.

$\check{c}[2] = [3\ 2\ 2\ 1] = [1\ 2\ 3\ 2] \Rightarrow 2$

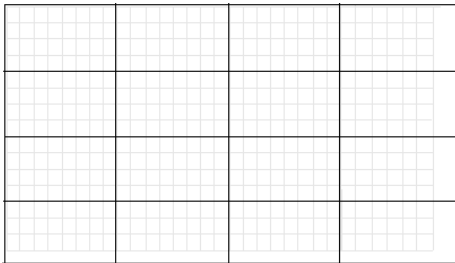
$\check{c}[3] = [2\ 1\ 2] = [1\ 2\ 2] \Rightarrow 2$

$\check{c}[4] = [1\ 2\ 8] = [1\ 2\ 8] \Rightarrow 2$

$\check{c}[5] = [2\ 8\ 9] = [2\ 8\ 9] \Rightarrow 8$

$\check{c}[6] = [8\ 9\ 9] = [8\ 8\ 9] \Rightarrow 9$

Distinct Block Operations



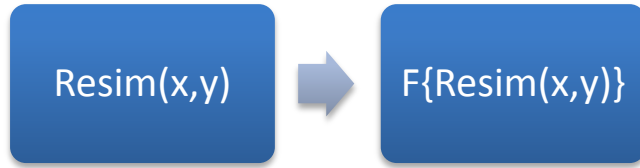
neighborhood operasyonunda olduğu gibi belirttiğimiz matris sıra ile resmi taramaz resim belirli bir sayıda parçaya ayrılır ve ardından her bölge için operasyon yapılır. neighborhood daki gibi matris resmin içinde kaymaz.

Frekans Tanım Bölgesi

Resim işlemedeki kullandığımız yöntemlerin yanı sıra resmin frekans tanım bölgesine geçmesi ile de bazı işlemler yaparız ve bu imge işlemede sıkça kullanılan yöntemlerdendir.

Daha önce filtreleme işlemi konvolusyon ile yapıyorduk. Frekans düzleminde ise çarpma işlemi ile yapıyoruz. Frekans domain'e geçiş ve geri dönüş işlemleri için hesapsal yük fazladır.

Frekans tanım bölgesine geçiş için fourier dönüşümü kullanılır.



Resim ayrık bir sinyaldir. Bundan dolayı fourier dönüşümü yapılırken ayrık zaman fourier dönüşümü kullanılır.

DFT(FFT):

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0, 1, \dots, N-1)$$

IDFT(IFFT):

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\left(\frac{2\pi}{N}\right)nk} \quad (n = 0, 1, \dots, N-1)$$

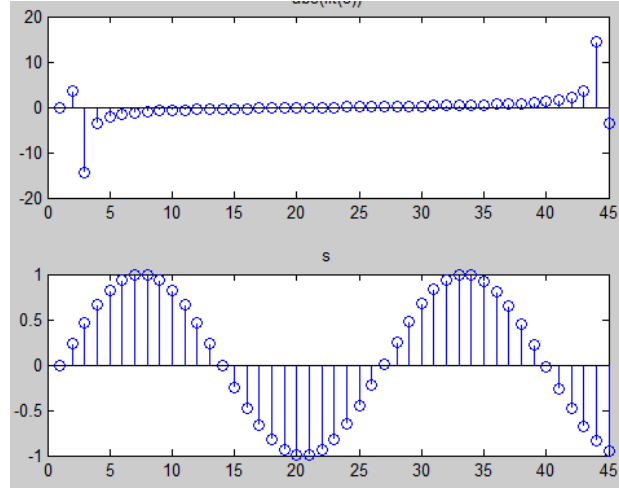
Matlab Komutu

```
fft2
2-D ayrık Fourier dönüşümü
Resim'in 2 boyutlu olması lazım.
Değer 1-D bir dizi dönüşümü yapılacak ise fft kullanılır.
```

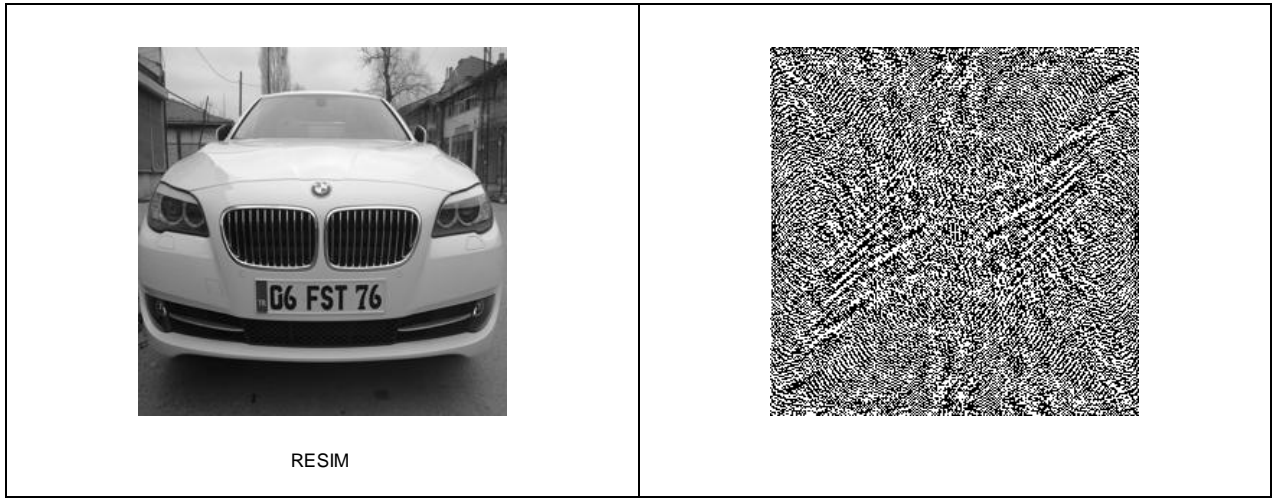
örnek

```
fs = 44100;
t = 0:1/fs:0.001;
s = sin(2 * pi * 1700 * t);
S=fft(s);
subplot(211), stem(S), title(' (fft(s)) ')
subplot(212), stem(s), title('s')
```

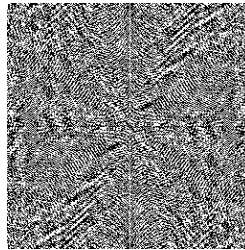
```
%ters fourier dönüşümü
a=ifft(S);
steam(a)
```

Bir resim'in fourier dönüşümü



Burada resim'in köşeleri 0 Hz ortalarına doğru gidildikçe frekans artar. fftshift komutu ile düşük frekanslar köşelerden ortaya kaydırılır. fftshift uygulandığında



Resim'in Yüksek Frekanslarını Alma Uygulaması(High Pass Filter)

Resim'in yüksek frekansı kısaca şu anlama gelir resimde detay çok ise bu resim yüksek frekanslı bir resimdir.

Eğer çok ayrıntılı olmayan bir resim ise bu düşük frekanslı bir resimdir.

```
Clear
I = imread('plaka.jpeg')
I_resize = imresize(I,[256 256])

I_gray = rgb2gray(I_resize);

figure
imshow(I_gray)
xlabel({'RESIM'});

F_I_gray = fft2(I_gray);
figure

F_I_gray = fftshift(F_I_gray);
imshow(F_I_gray)
%*****mask*****

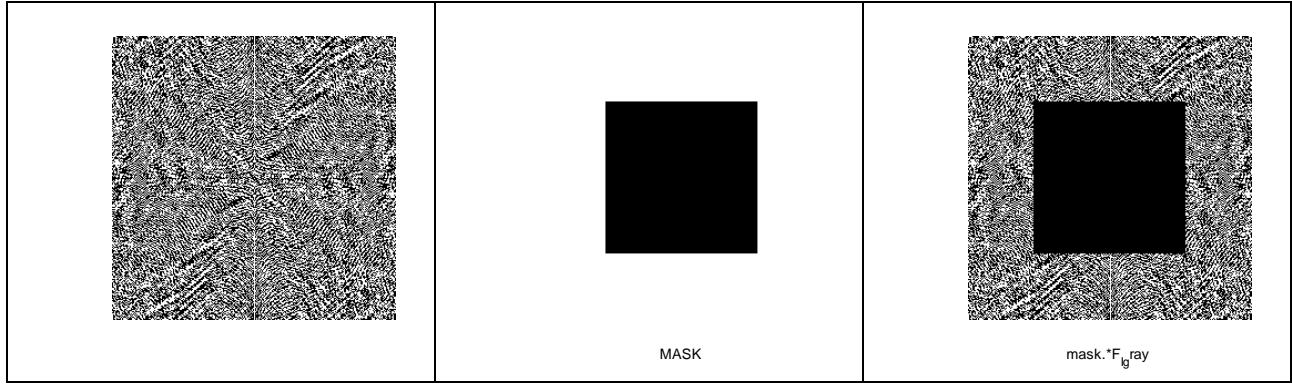
mask1 = ones(256,256);
mask = zeros(256,256);
mask(30:226,30:226) = 1;
mask = mask1 - mask;
%mask = imread('open_app.bmp');
%mask = imresize(mask,[256 256]);
%mask = ones(256,256) - mask;

figure
imshow(mask)
xlabel({'MASK'});

figure
imshow(mask.*F_I_gray)
xlabel({'mask.*F_I_gray'});

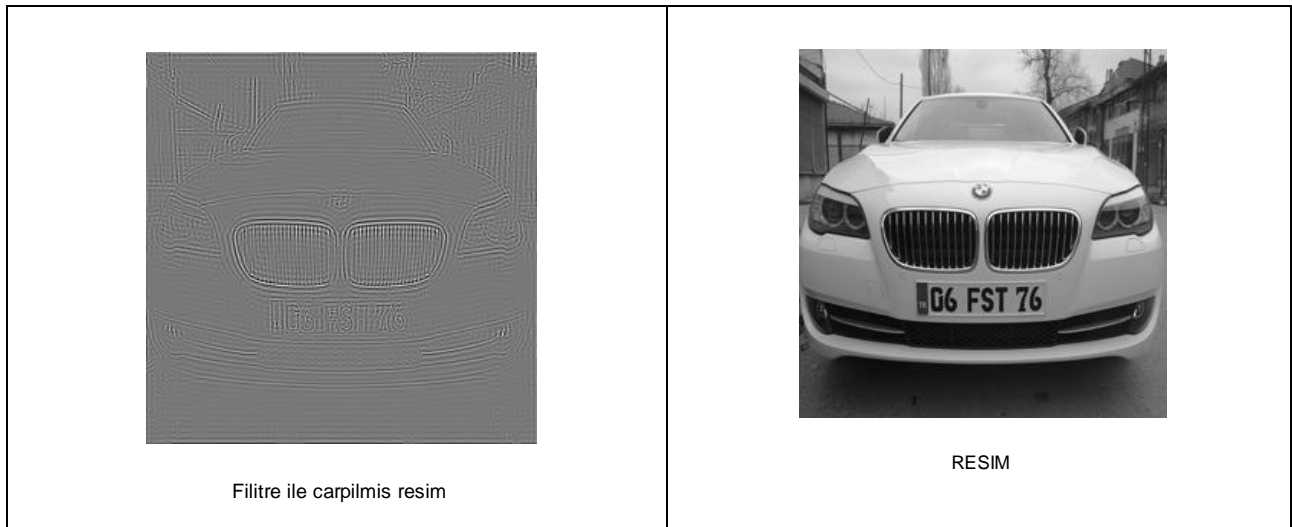
F_I_gray = mask.*F_I_gray;
F_I_gray = fftshift(F_I_gray);
figure
imshow(F_I_gray)
resim2 = ifft2(F_I_gray);

figure
imshow(resim2,[])
xlabel({'Filtre ile carpilmis resim'});
```



şimdi 3.resmin inverse fourierini aldığım da resmin yüksek frekanlı kısımları kalacak

```
resim2 = ifft2(F_I_gray);  
%kodu ile inverse fourier transform yapılır.
```

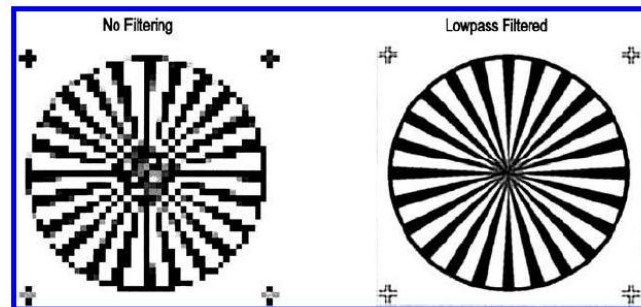
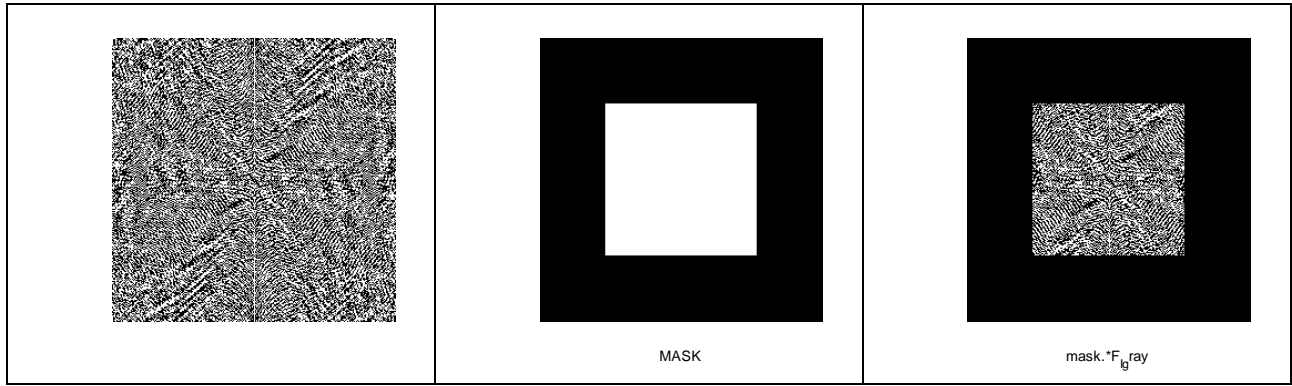


Yüksek frekans resimdeki ayrıntılardır. Arabanın ön ızgaraları çok ayrıntılı filtrelenmiş resimde de oralar gözükmektedir.

Resim'in Düşük Frekanslarını Alma Uygulaması(Low Pass Filter)

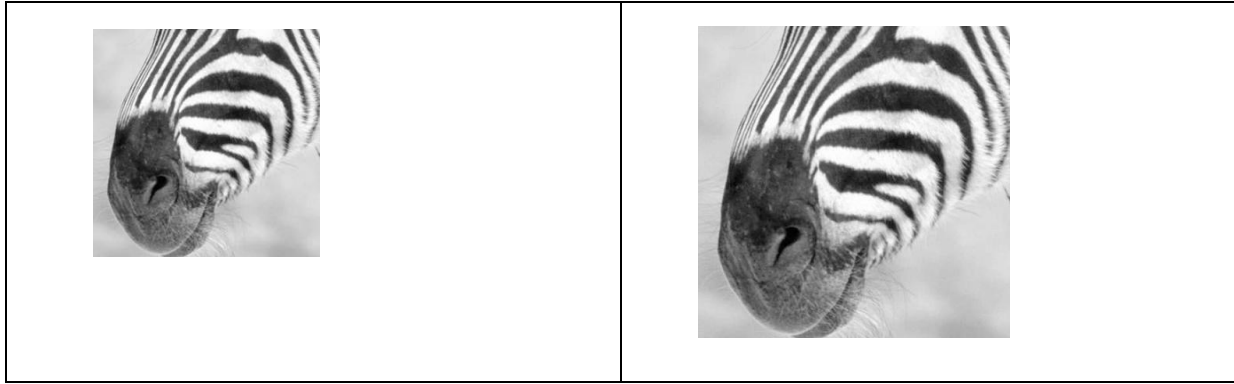
Sadece maskeyi değiştireceğim. Eski filtre kenarları alıyordu şimdi ise kenarları almayacak. Sadece orta kısmı alacak yani düşük frekanslı ayrıntısı az olan kısmı.

```
mask = zeros(256,256);  
mask(60:196,60:196) = 1;
```

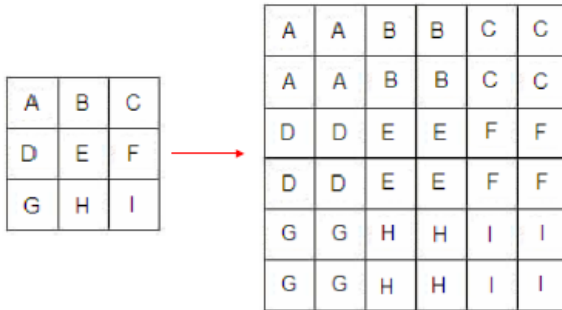


Resim Boyutlandırma
imresize komutu ile yapılır

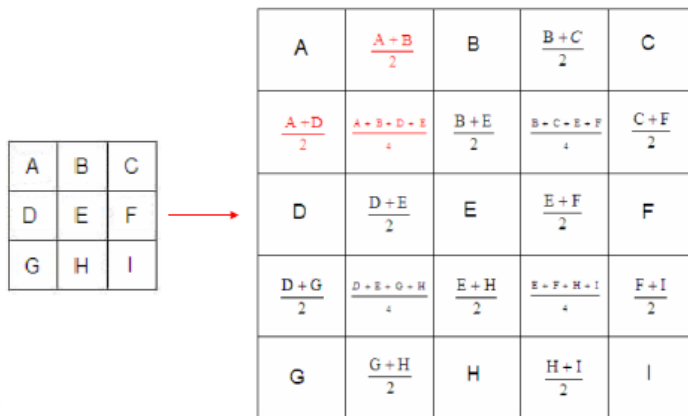
```
I = imread('plaka.jpeg')
I_resize = imresize(I,[256 256])
```

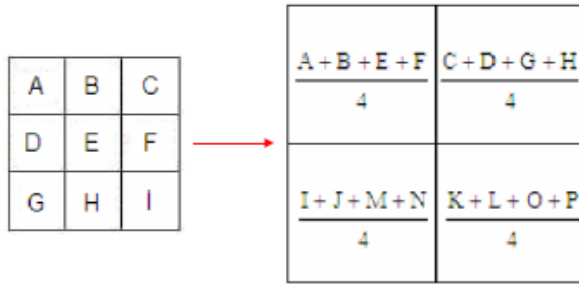


- Yakınlaştırma, düşük piksel boyutlu bir imgenin piksel boyutunun yazılımsal olarak artırılmasıdır.
- Sayısal yakınlaştırma (digital zoom).



- Boyut büyültmede daha yumuşak geçişler için:





Boyut küçültme işlemi

Resim Kırpma

imcrop komutu ile mouse ile resmin istediğimiz yeri kırpamızı sağlar.

Resim kırpma matrisin istediğimiz kısmını almamız demektir o zaman bu komutu kullanmadanda aşağıdaki gibi resim'i kırpa biliriz.

```
Clear
I = imread('lena_bw.png');
I_crop = I(200:300 , 200 : 300);
```

komutu ilede yapılır.



Resmin Yönü

```
I_rotate = imrotate(I, deg, method, bbox);
```

Resmin yönünü ayarlamamıza yarar.

Resim Bölütleme

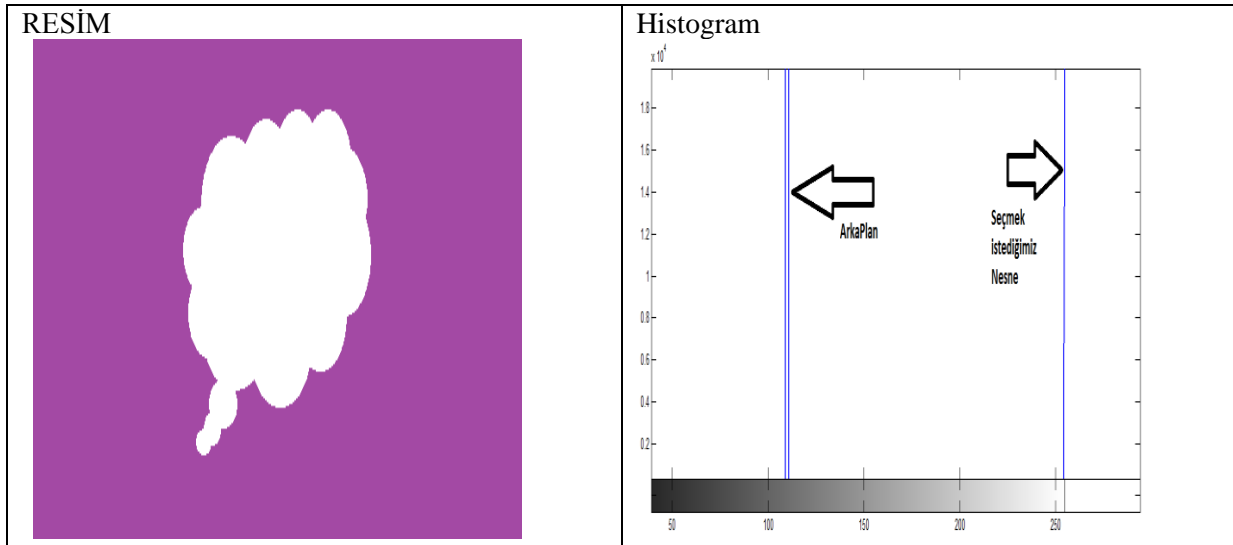
- 1) Thresholding methods such as [Otsu's method](#)
- 2) Color-based Segmentation such as [K-means clustering](#)
- 3) Transform methods such as [watershed segmentation](#)
- 4) Texture methods such as [texture filters](#)

Resim bölütlemeye dört temel yöntem var ama ben bunlardan sadece ilk ikisi üzerine çalışmalar gerçekleştirdim.

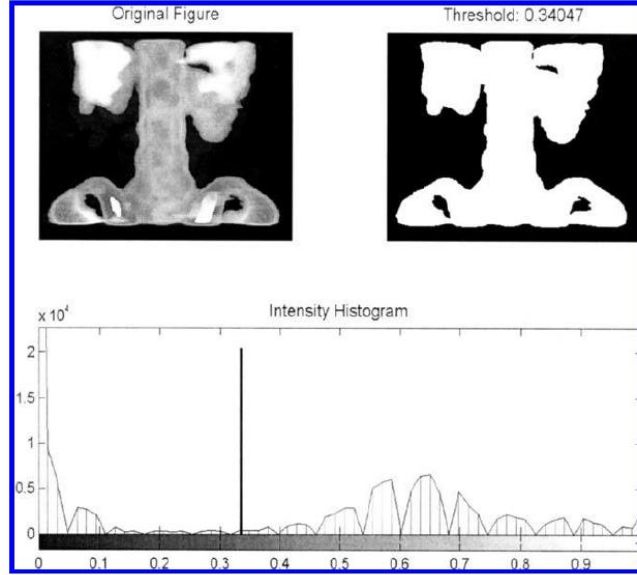
Bölütleme temel resim işlemlerine hakim olmam amacı ile belirlenen bir konudur. Aslında bölütleme hala üzerine araştırmalar yapılan hala tam olarak çözülmemiş bir sorundur ve bu iki yöntemi bildiğimiz takdirde bölütleme konusuna hakim olabiliyoruz.

1)Thresholding methods such as Otsu's Method

Bu yöntemi basitçe anlatmak gerekir ise beyaz bir kağıdın üzerinde siyah bir daire olsun beyaz kısmı çıkarttığımız zaman sadece daire kalır. Eşikleme metodunda belirlenen bir eşik değerinin üstü 1 altı 0 olarak belirlenir. Ayrıca bu yöntem de histogram dan yararlanılır. Histogramdaki en yüksek tepe noktasına sahip renk arka planı oluşturur diğer tepeler ise ayrı ayrı nesneleri oluşturur. Eşik değerini belirlerken bunlara dikkat etmeliyiz siyah renk bir zeminin üzerindeki siyah telefonun resminin eşikleme işlemi yaptığımızda telefonda ortadan kaybolmuş olur bu bundan dolayı bu yöntemde resmin arka planı önemlidir.

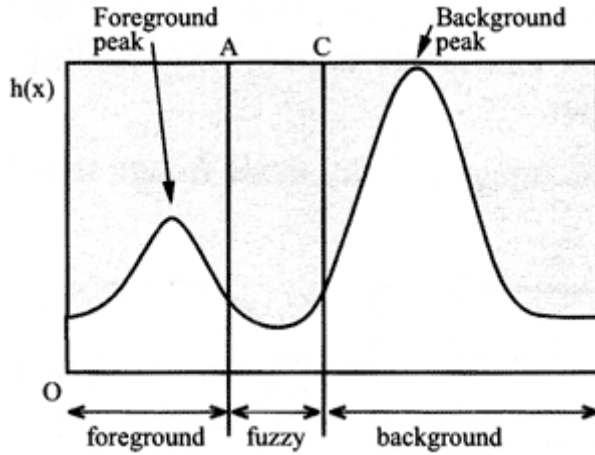


Histogram'a bakarak arkaplan 0 olacak şekilde threshold işlemini gerçekleştir.



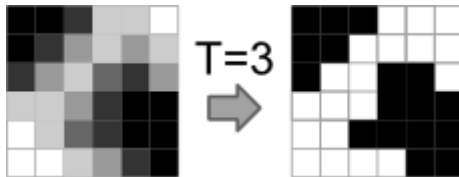
Örnekte de görüldüğü gibi röntgende Threshold değeri 0.34047 değerini seçtik threshold işlemi uygulanmış olan resim 0.34047 nin üstündeki yerler 1 geri kalan 0 olarak resim thresholdlandı.

Ama her zaman histogram a bakıp eşik değerini el ile ayarlayamayız. Bilgisayara bunu otomatik olarak yaptırmamız gerekir bunu için Otsu's yöntemi vardır..



Otsu yöntemi şekil deki gibi histogramı max olduğu yeri background olarak belirler ve diğer tepe noktasını nesne olarak belirler ve resimdeki fuzzy aralığında bir değerde eşik değerimizi oluşturur.

Örnek:



Otsu Algoritması

Within Class Variance $\sigma_W^2 = W_b \sigma_b^2 + W_f \sigma_f^2$ (as seen above)

Between Class Variance $\sigma_B^2 = \sigma^2 - \sigma_W^2$
 $= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2$ (where $\mu = W_b \mu_b + W_f \mu_f$)
 $= W_b W_f (\mu_b - \mu_f)^2$

Matlab graythresh fonksiyonunun p, omega, mu , mu_t yi bulduğu kısmı

```
I = im2uint8(I(:));
num_bins = 256;
counts = imhist(I,num_bins);

p = counts / sum(counts);
omega = cumsum(p);
mu = cumsum(p .* (1:num_bins)');
mu_t = mu(end);

previous_state = warning('off', 'MATLAB:divideByZero');
sigma_b_squared = (mu_t * omega - mu).^2 ./ (omega .* (1 -
omega));
warning(previous_state);

% Find the location of the maximum value of sigma_b_squared.
maxval = max(sigma_b_squared);
isfinite_maxval = isfinite(maxval);
if isfinite_maxval
    idx = mean(find(sigma_b_squared == maxval));
    % Normalize the threshold to the range [0, 1].
    level = (idx - 1) / (num_bins - 1);
else
    level = 0.0;
end
else
    level = 0.0;
end

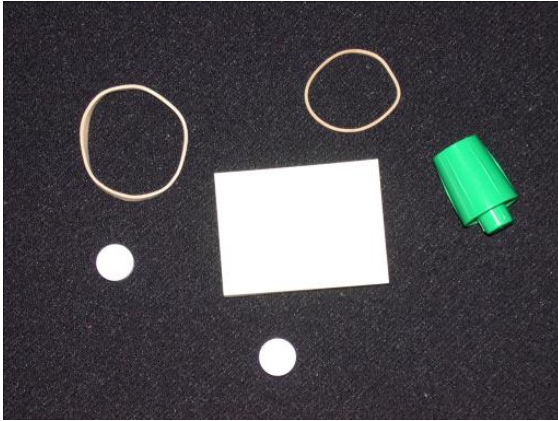
% compute the effectiveness metric
if nargin > 1
    if isfinite_maxval
        em = maxval / (sum(p .* ((1:num_bins).^2)') - mu_t^2);
    else
        em = 0;
    end
end
```

Otsu yöntemi ile eşik değeri bulma matlab da graythresh fonksiyonu ile kolayca bulunur.

Örnek

```
I = imread('coins.png');  
level = graythresh(I);  
BW = im2bw(I,level);  
imshow(BW)
```

Threshold yöntemi ile yapılmış bir uygulama.



Orjinal Resim

Step 1: Resimi alma

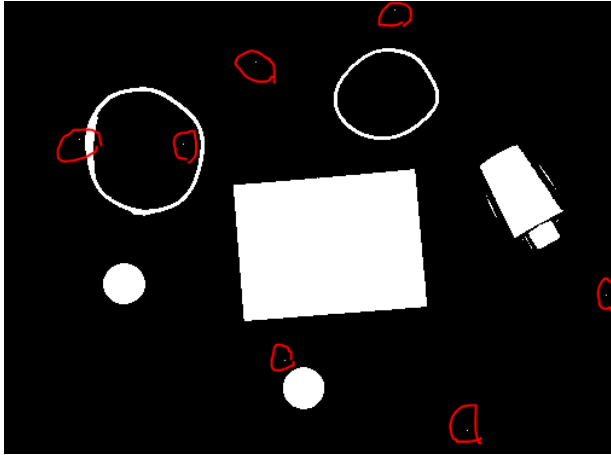
```
RGB = imread('pillsetc.png');  
imshow(RGB);  
I = rgb2gray(RGB);
```

resim üzerinde işlemler yapabilmemiz için resmi gri seviyeli bir resime dönüştürdüm.

Step 2: Resimin eşik değerini belirleme.

graythresh komutu ile belirlediğimiz eşik değeri ile resmi siyah beyaz hale dönüştürür.

```
threshold = graythresh(I);  
bw = im2bw(I,threshold);  
imshow(bw)  
% remove all object containing fewer than 30 pixels  
bw = bwareaopen(bw,30);
```

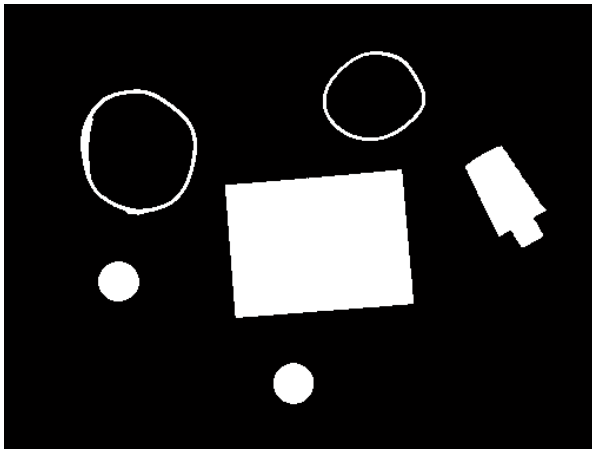


Step 3: Remove the Noise

```
bw = bwareaopen(bw,30);  
% fill a gap in the pen's cap  
se = strel('disk',2);  
bw = imclose(bw,se);
```

Yukarıdaki şekilde görüldüğü gibi kırmızı ile işaretlenen yerler gürültü(noise) bunları kaldırmamız lazım çünkü objeleri sayarken etrafı siyah ile çevrili her piksel topluluğu bir nesne olarak değerlendirilebilir buda sayım işlemimizin yanlış olmasına neden olur.

Burada yapılan işlem aslında bir morfolojik bir işlemdir.



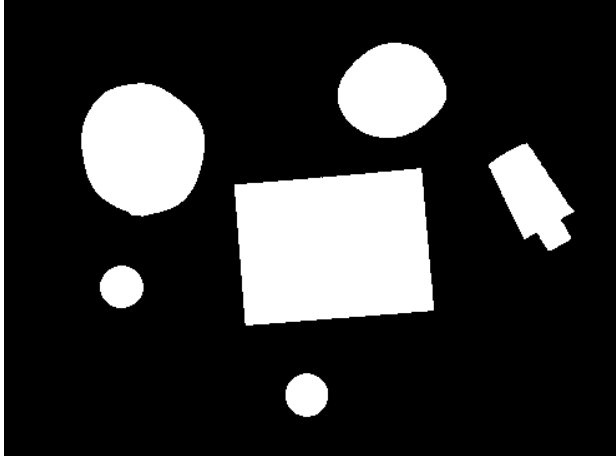
Step 4: Fill Hole

Bunu amacı objeyi en dıştan seçebilmek içindir.

```
bw = imfill(bw,'holes');  
cc = bwconncomp(bw,4)  
imshow(bw)
```

```
cc =  
    Connectivity: 4  
    ImageSize: [384 512]  
    NumObjects: 6  
    PixelIdxList: {1x6 cell}
```

Resmimizde 6 tane obje olduğunu belirledik

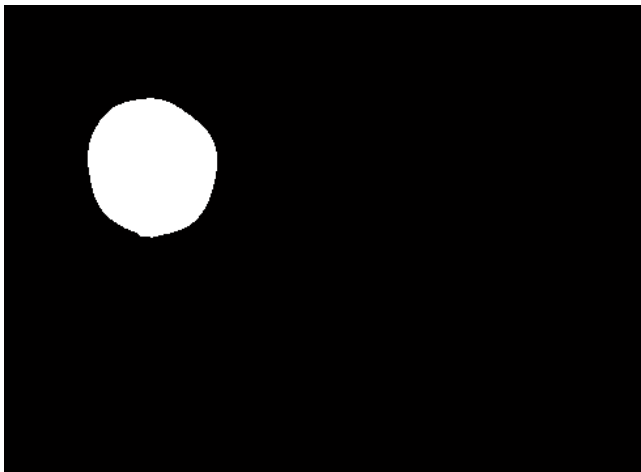


```
obje = false(size(bw));  
obje(cc.PixelIdxList{1}) = true;  
imshow(obje);
```

Bu kodum ile de 1.objeyi görüntülüyorum

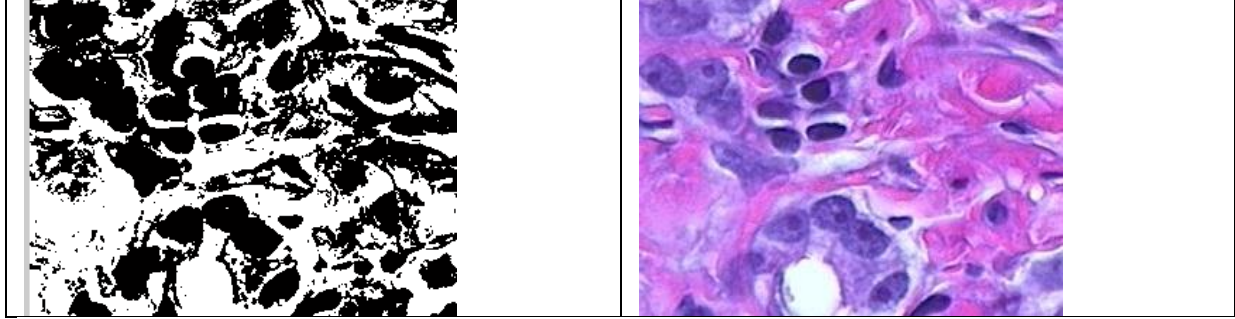
obje(cc.PixelIdxList{1}) = true; komutunun içindeki 1 indisini değiştirdiğimiz zaman girdiğimiz değerın objesini gösterir

Aşağıdaki şekilde 1. nesne gösterilmiştir.



Color-based Bölütlemesuch as K-means clustering

Threshold yöntemi genel olarak kullanılan yöntem olsa da arka planın değişik renklerde olduğu yerler bu yöntemi kullanamayız bunun için renk tabanlı bölütleme yapmalıyız.



Orijinal resim'e eşikleme yöntemi uygulandığımız zaman yukarıda da görüldüğü gibi beyaz renkte pembe renkte beyaz renk.

Bu siyah beyaz resimden mavi ve pembe renkleri çıkarma imkanımız yok bu sebepten dolayı Renk tabanlı bölütleme kullanmamız gerekiyor. Bunu yaparken de K-means clustering yöntemini kullanmamız gerekir.

En eski kümeleme algoritmalarından olan k-means,1967 yılında J.B. MacQueen tarafından geliştirilmiştir (MacQueen, 1967) *

Algoritma istatistiksel olarak benzer nitelikteki değerleri aynı gruba sokar. Bir elemanın yalnızca bir kümeye ait olmasına izin verir. Küme merkezi kümeyi temsil eden değerdir.

Burada en önemli iki amaç şudur:

- 1- Küme içindeki değerler birbirlerine en çok benzemeli,
- 2- Kümeler birbirine mümkün olduğunca benzememeli

$$J = \sum_{j=1}^k \sum_{i=1}^n \left\| x_i^{(j)} - c_j \right\|^2$$

H&E image

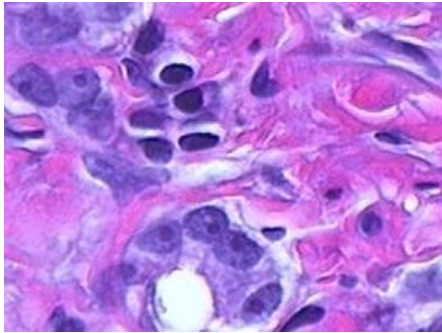
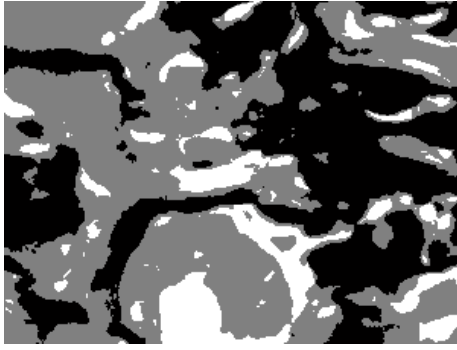


Image courtesy of Alan Partin, Johns Hopkins University

Resimde görüldüğü gibi belirli bir rengin baskınlığı yok ama biz maviye , pembeye , beyaza yakın olan yerleri ayrı ayrı almak istiyoruz.



K-means yöntemi ile resmi 3 küme şeklinde şekline çevirdik bundan sonrası artık çok kolay beyazı istiyorsak 2. resimden beyazı seçip 1. resim ile çarptığımız zaman direk istediğimiz bölüm çıkacaktır.

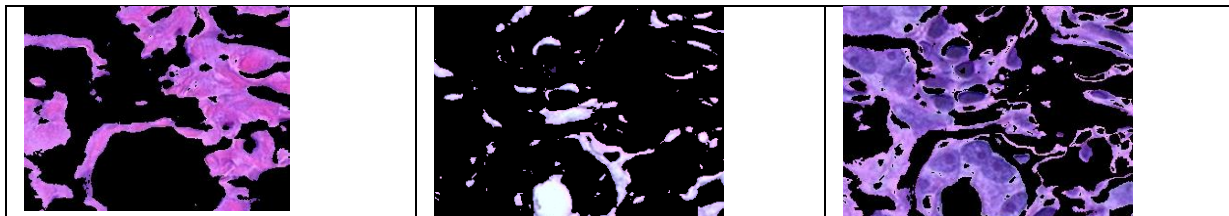
```
nColors = 3;
% repeat the clustering 3 times to avoid local minima
[cluster_idx, cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean','Replicates',3);

pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster index');

segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);

for k = 1:nColors
    color = he;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
end
```

```
imshow(segmented_images{1}), title('objects in cluster 1');
figure
imshow(segmented_images{2}), title('objects in cluster 2');
figure
imshow(segmented_images{3}), title('objects in cluster 3');
```



Görüldüğü gibi resim resimi oluşturan 3 renke bölütlendi.

Edge Detection (Ayrıt Yakalama)

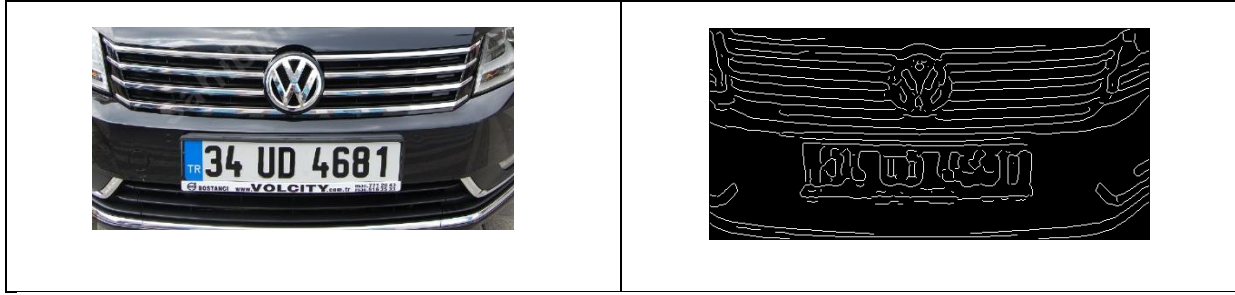
Görüntülerin içindeki nesnelerin ayrıtlarını bulmak için kullanılan bir görüntü işlem tekniğidir. Kesikliklerin tespit ederek çalışır.

Kenar algılama ;

Görüntü bölütleme gibi görüntü işleme, bilgisayar görme, ve makine görme gibi alanlarda ve veri çıkarmak için kullanılır.



Örnek bir ayrıt algılama



Matlab Kodu

```
I = imread('plakal.png');  
imshow(I)  
I = rgb2gray(I);  
I = im2double(I);  
h = fspecial('gaussian',12,12); % Construct gaussian  
%gaussian filtresinin amacı resmi bulanıklaştırmak ve ayrıt yakalam  
%algoritmasının ayrıntılara takılmaması sağlanır.  
  
I_f = imfilter(I,h,'replicate'); % Filter image  
I_edge = edge(I_f,'canny',.3); % To remove edge  
figure  
imshow(I_edge)
```

Algoritması

Bu yöntem türev kullanılarak yapılıyor ayrık zamanda türev işlemi çıkarmaya karşılık gelir.

Mesela Beyaz kağıdın üzerindeki yuvarlak tam beyazdan siyaha geçtiği yerde $1 - 0 = 1$ program buradan anlıyor farklı bir yere geçtiğini ve o geçtiği noktayı sınır olarak belirliyor.

Bölütleme ile ilgili kendi yaptığım çalışma

Bu kodu farklı resimlerde test ettim sonuçlar aşağıda.

```
RGB = imread('iphone.jpg');
%RGB = imread('ex3.jpg');
imshow(RGB);
I = rgb2gray(RGB);
threshold = graythresh(I);
bw = im2bw(I,threshold);

% remove all object containing fewer than 30 pixels
if bw(1,1) == 1
    bw = imcomplement(bw);
end
bw = bwareaopen(bw,30);

bw = bwareaopen(bw,30);
% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);

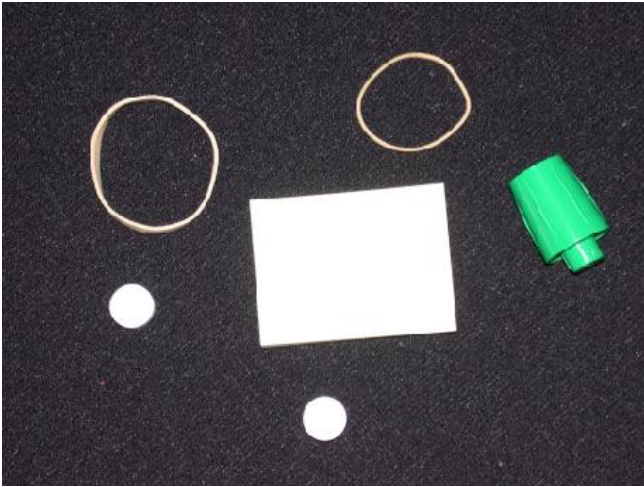
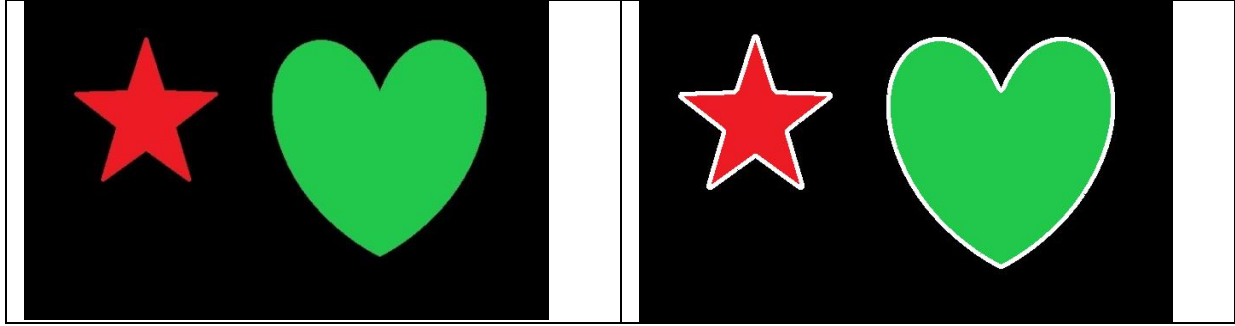
[B,L] = bwboundaries(bw,'noholes');

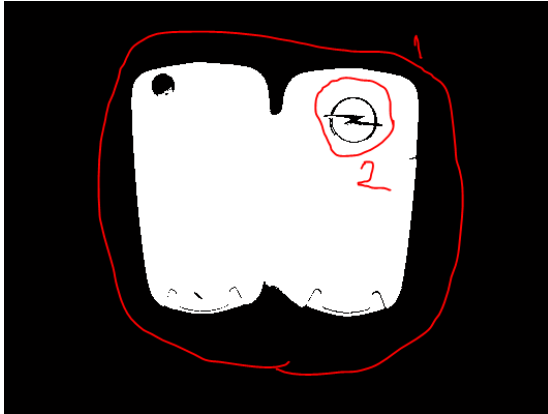
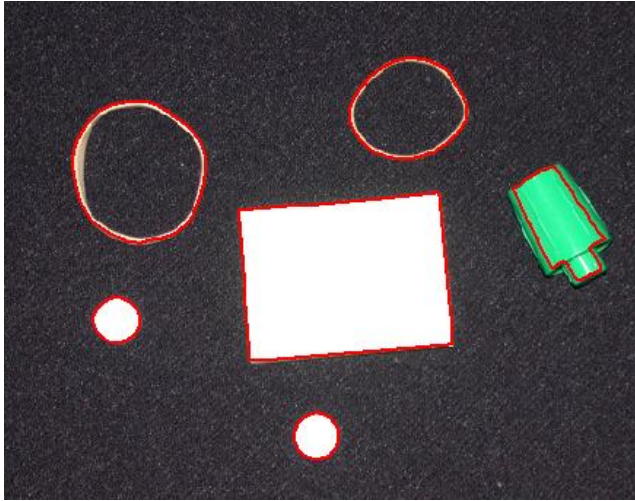
% Display the label matrix and draw each boundary

hold on
for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2)
end
cc = bwconncomp(bw)
%%cc = bwconncomp(bw,4)
%imshow(bw)

obje = false(size(bw));
obje(cc.PixelIdxList{1}) = true;
%%imshow(obje);
```


Calisma Sonuclari





Resimde 2 tane anahtar var

cc =

```
Connectivity: 8
ImageSize: [2448 3264]
NumObjects: 2
PixelIdxList: {[2102408x1 double] [20089x1 double]}
```

Resimlerde gözüktüğü gibi 2 tane obje olduğunu söylüyor ama bunlar bizim istediğimiz objeler değil buradaki sorunu çözmek için morfolojiden yararlanınız..

Morfoloji

Biyolojinin canlıların şekil ve yapıları ile ilgilenen dalına morfoloji (biçim bilim) adı verilmektedir. Matematiksel morfoloji ise temel küme işlemlerine dayanan, imgedeki sınırlar (borders), iskelet (skeleton) gibi yapıların tanımlanması ve çıkartılması, gürültü giderimi, bölütleme gibi uygulamalar için gerekli bir araçtır. İmge işlemede genellikle, morfolojik süzgeçleme, inceltme (thinning), budama (pruning) gibi ön/son işlem olarak da sıkça kullanılırlar. Gri tonlu imgeler üzerinde de yapılabileceği gibi, genellikle ikili imgeler üzerinde yapılan işlemlerdir.

morfoloji'nin Kullanım alanlar

- Görüntü geliştirme
- Görüntü bölütleme
- Görüntü onarma
- Kenar yakalama
- Doku analizi
- Parçacık Analizi
- Genelleştirme
- İskelet belirleme
- Şekil analizi
- Görüntü Sıkıştırma
- Bileşen analizi
- Eğri keskinleştirme
- İnceltme
- Özellik ayırma
- Gürültü azaltma
- Boşluk azaltma

6. Projenin Biten Kısımları Ve Gelecek Çalışmalar

Projenin amacı morfolojik işlemlerle belirlenen bir sorunu çözmek idi örnek sorunlar mobese kameralarından araçların plakasını tespit etme , fabrikalardaki kalite kontrol sistemleri , tıp alanında kanserli hücrelerin tespiti ve bir çok işlem. Morfolojik işlerin kapsamına girmektedir. Tabi bu işlemlerin yapıla bilmesi için önce temel resim işlemlerinin öğrenilmesi gerekmektedir.

Bundan dolayı projemizin birinci dönemini temel resim işlemleri ve matlab kullanılması, ikinci bölümünü de morfolojik işlemler ile belirlenen bir sorunu çözme örneğin bir arabanın plakasını tespit etme gibi.

Projenin birinci dönem hedefi olan temel resim işlemlerinin ve matlab kullanımının geliştirilmesi. Belirlenen kaynaklar okunarak temel resim işlemleri (Boyutlandırma , renk dönüşümü , frekans tanım bölgesinde işlemler ve resim bölütleme) öğrenildi ve matlab'da kendi yaptığım uygulamalar ve dökümanlardaki örnekler ile matlab kullanımı geliştirildi.

Projemizin ikinci dönem hedefinde ise birinci dönem öğrendiğim ve ikinci dönemde öğreneceğim morfolojik işlemler ile birlikte belirlenen bir sorunu çözmeye çalışacağım

7. Kaynakça

[1]Digital Image Processing (3rd Edition) August 31, 2007

[2] Copyright 2004 by Marcel Dekker, Inc. All Rights Reserved.

[3] <http://www.mathworks.com/>