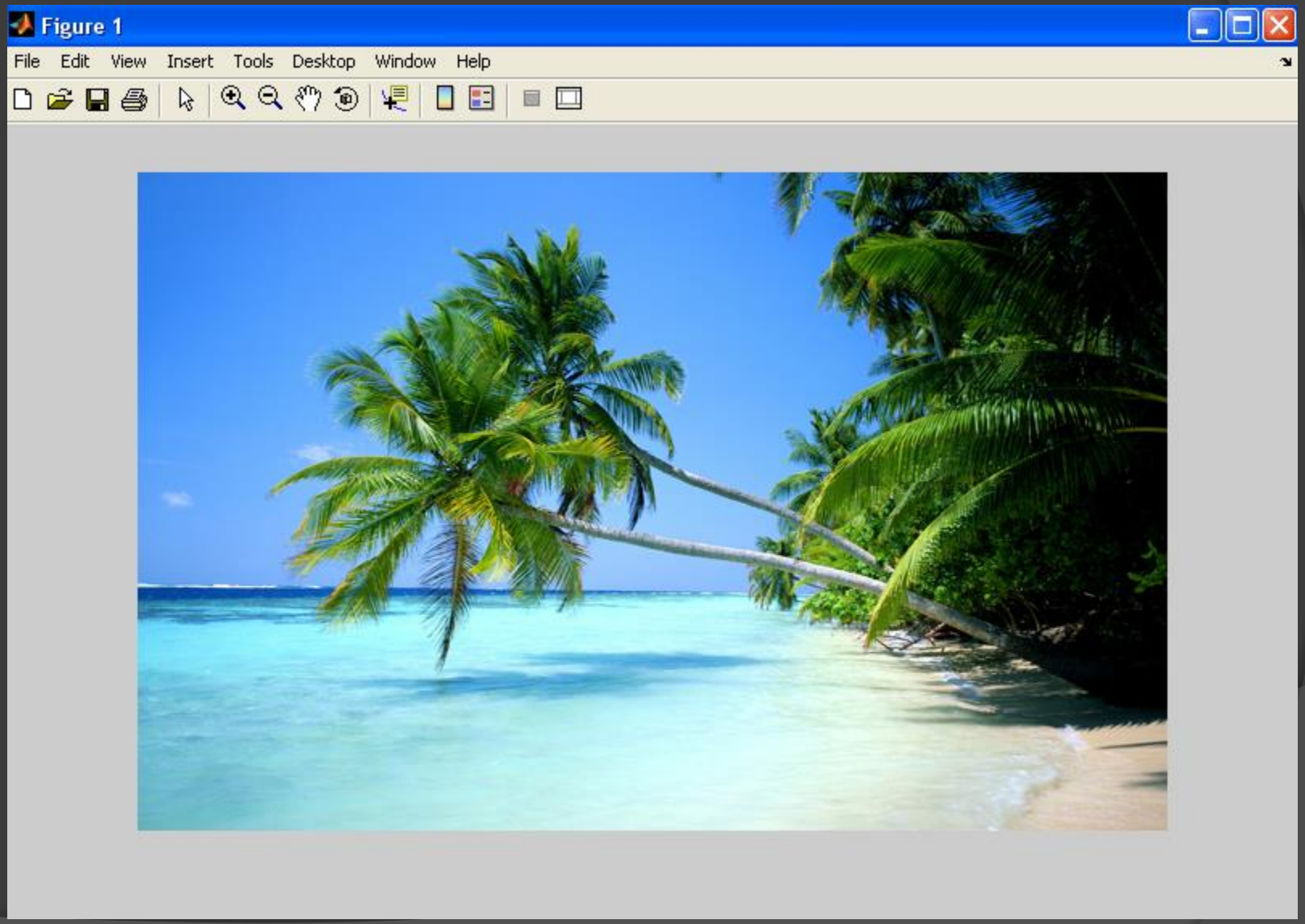
Bilgisayar Görmesi

Ders 4: İSTATİSTİKSEL İŞLEMLER MATLAB UYGULAMALARI

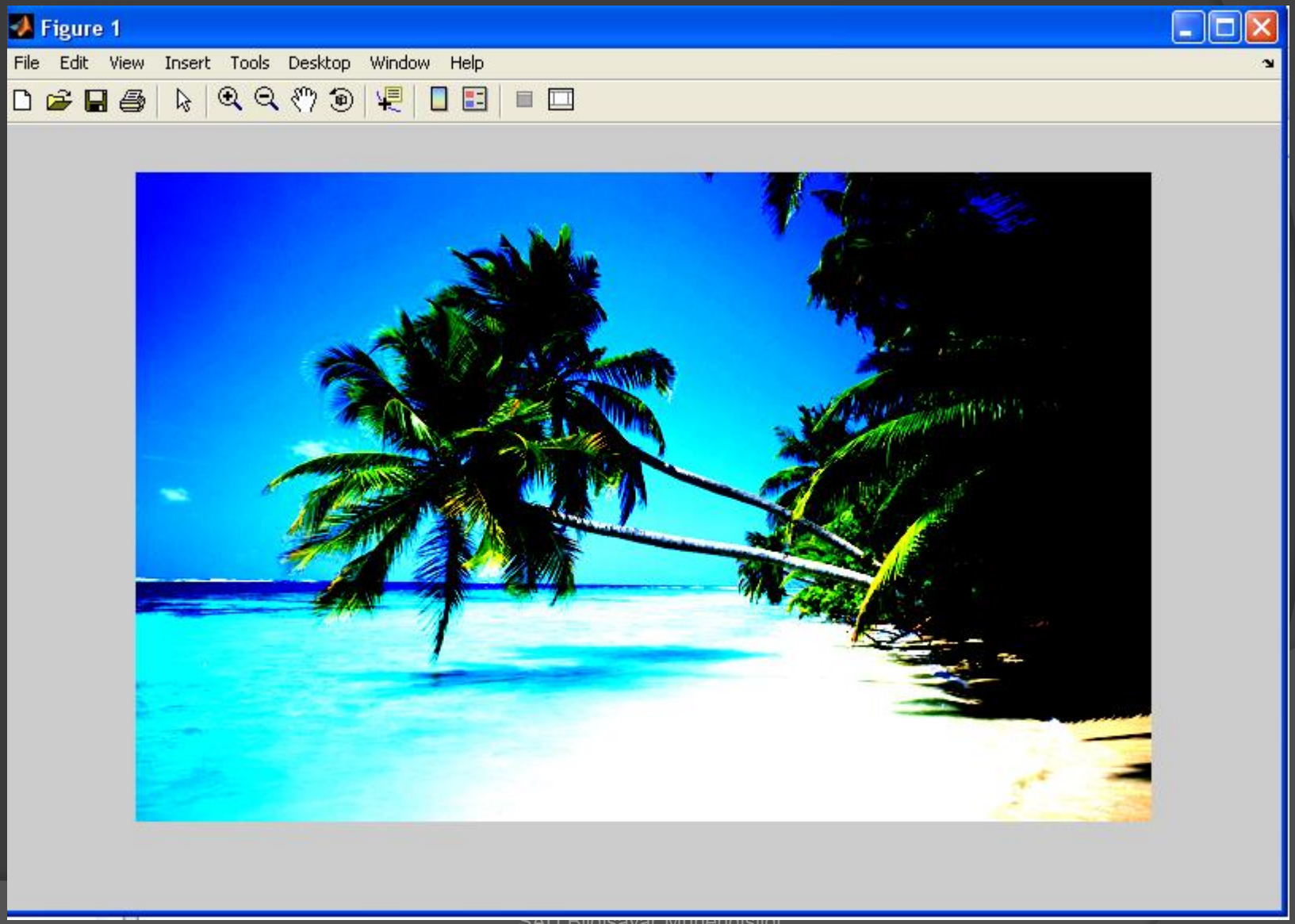
Dr. Öğr. Üyesi Serap ÇAKAR

```
f=imread('c:\r2.jpg');
İmshow(f);
```

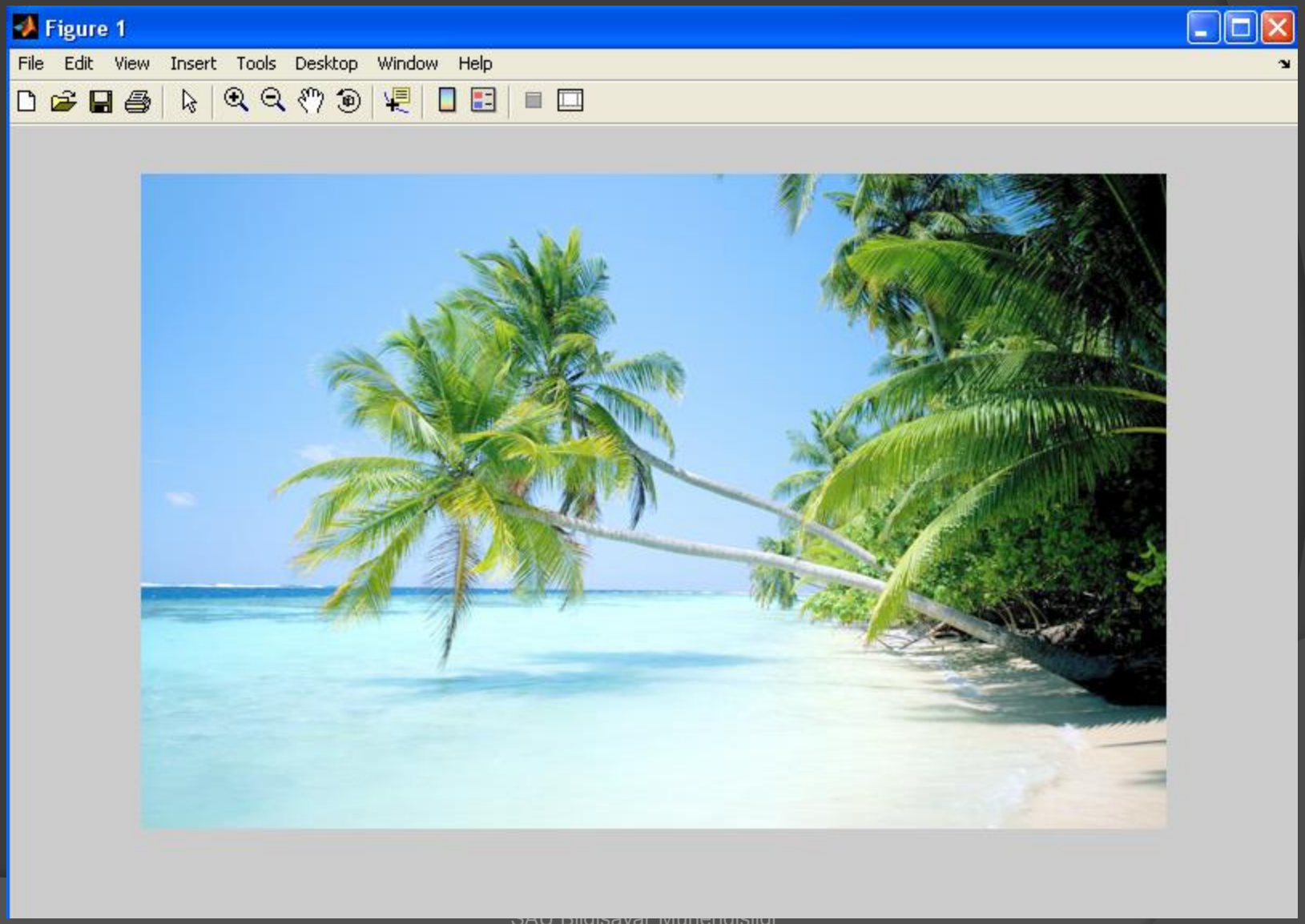


G=imadjust(f, [low_intensity High_intensity] [low_out High_out], gamma)

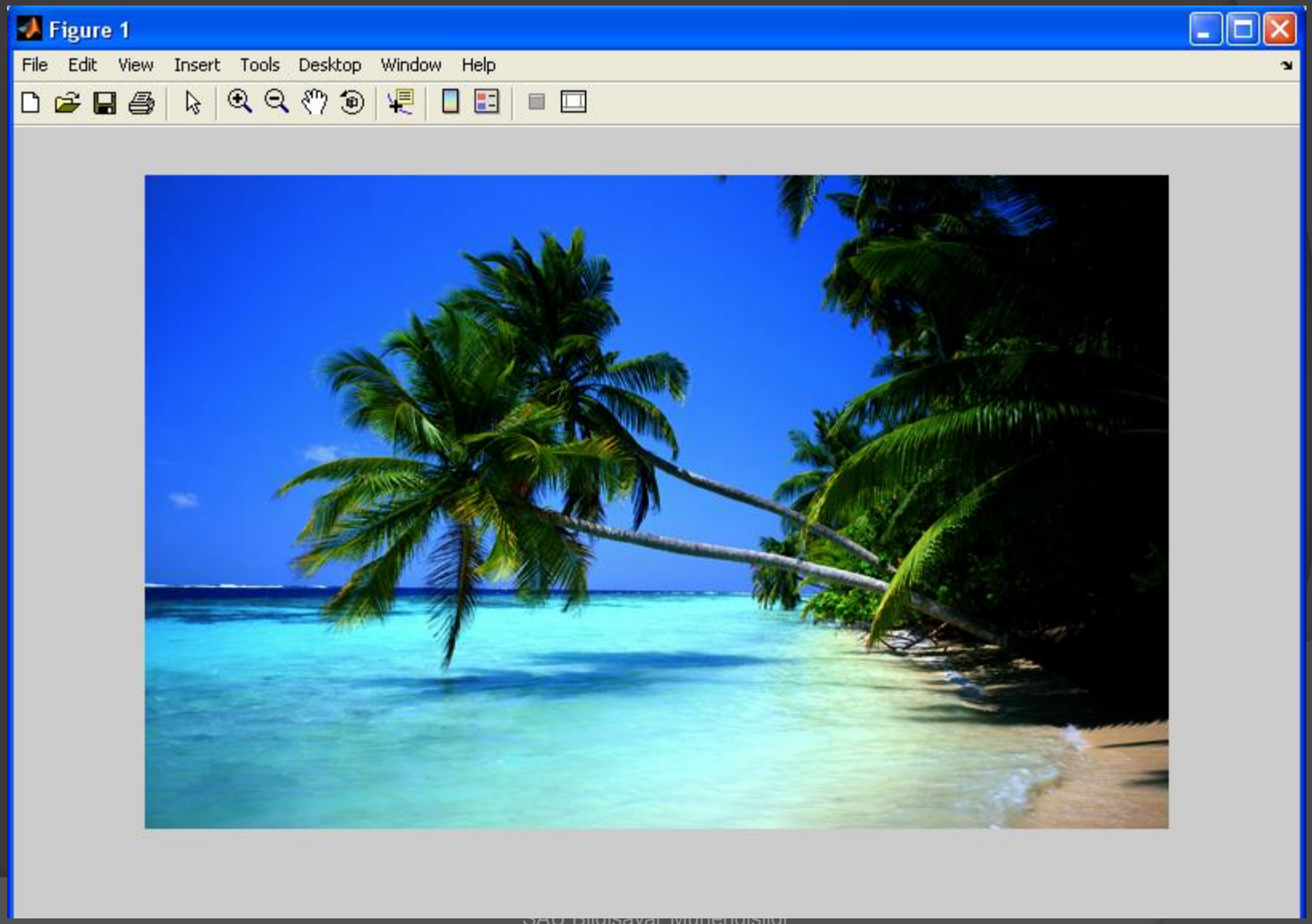
g=imadjust(f, [0.5 0.75], [0 1]); (gamma kullanılmadığında1 olarak ayarlanır)



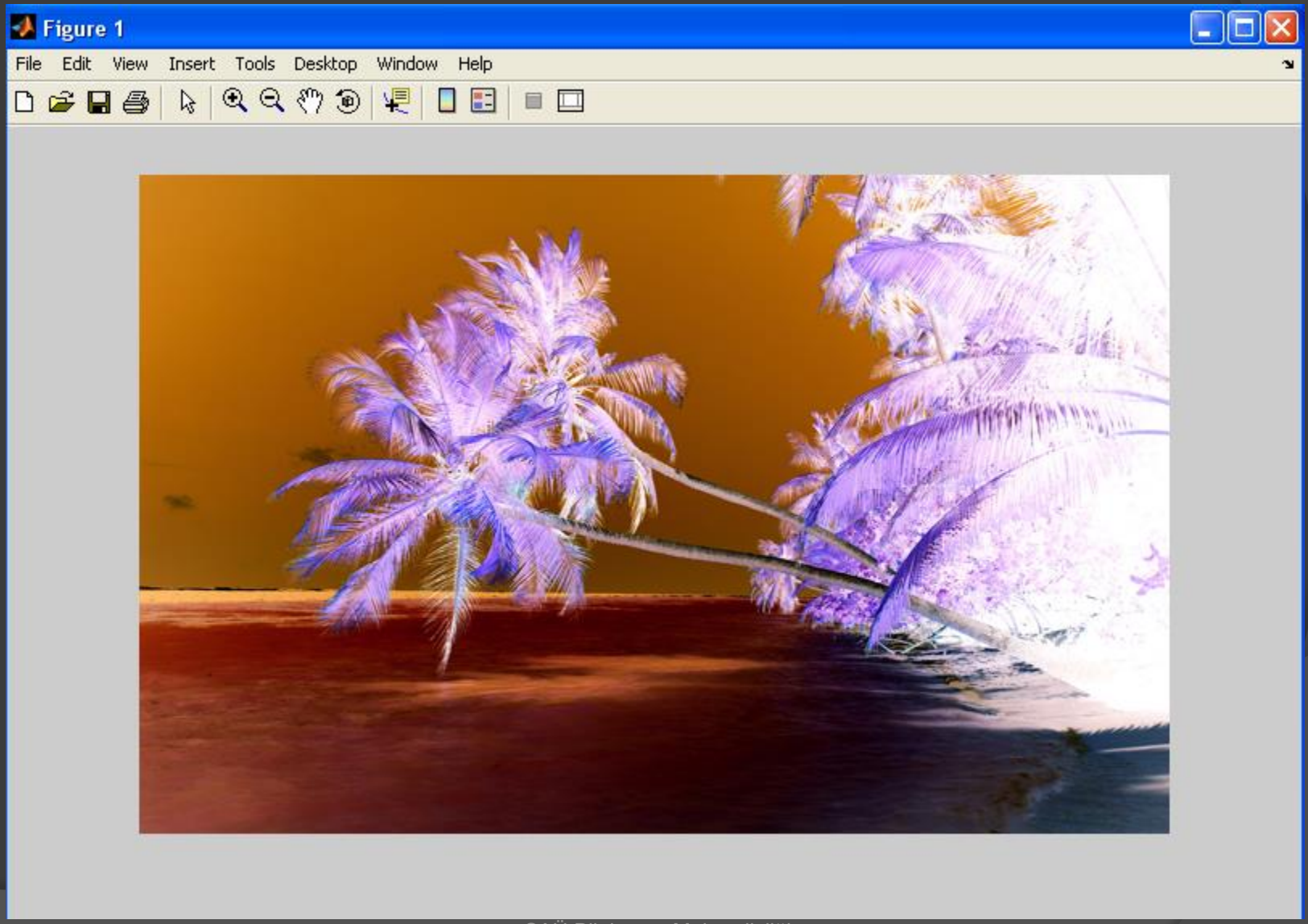
Eğer gamma 1'den küçükse, çıkış değerleri yükselir ve görüntü parlaklaşır.
g=imadjust(f, [0 1], [0 1],0.5);



Eğer gamma 1'den büyükse, çıkış değerleri azalır ve görüntü koyulaşır.
g=imadjust(f, [0 1], [0 1],2);

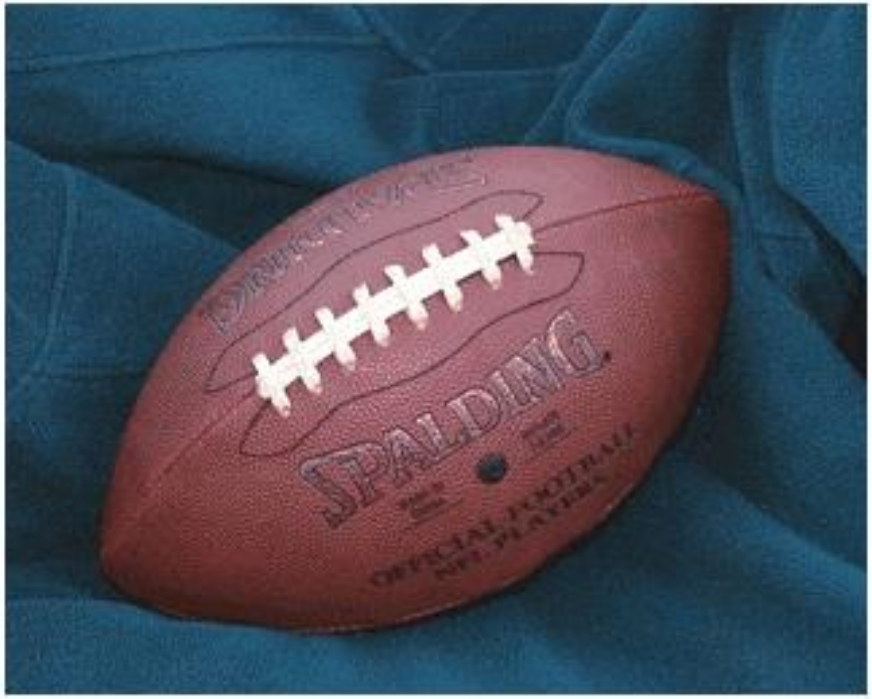


Eğer high_out < low_out, çıkış görüntüsü negatif görüntüye dönüşür.
g=imadjust(f, [0 1], [1 0]);

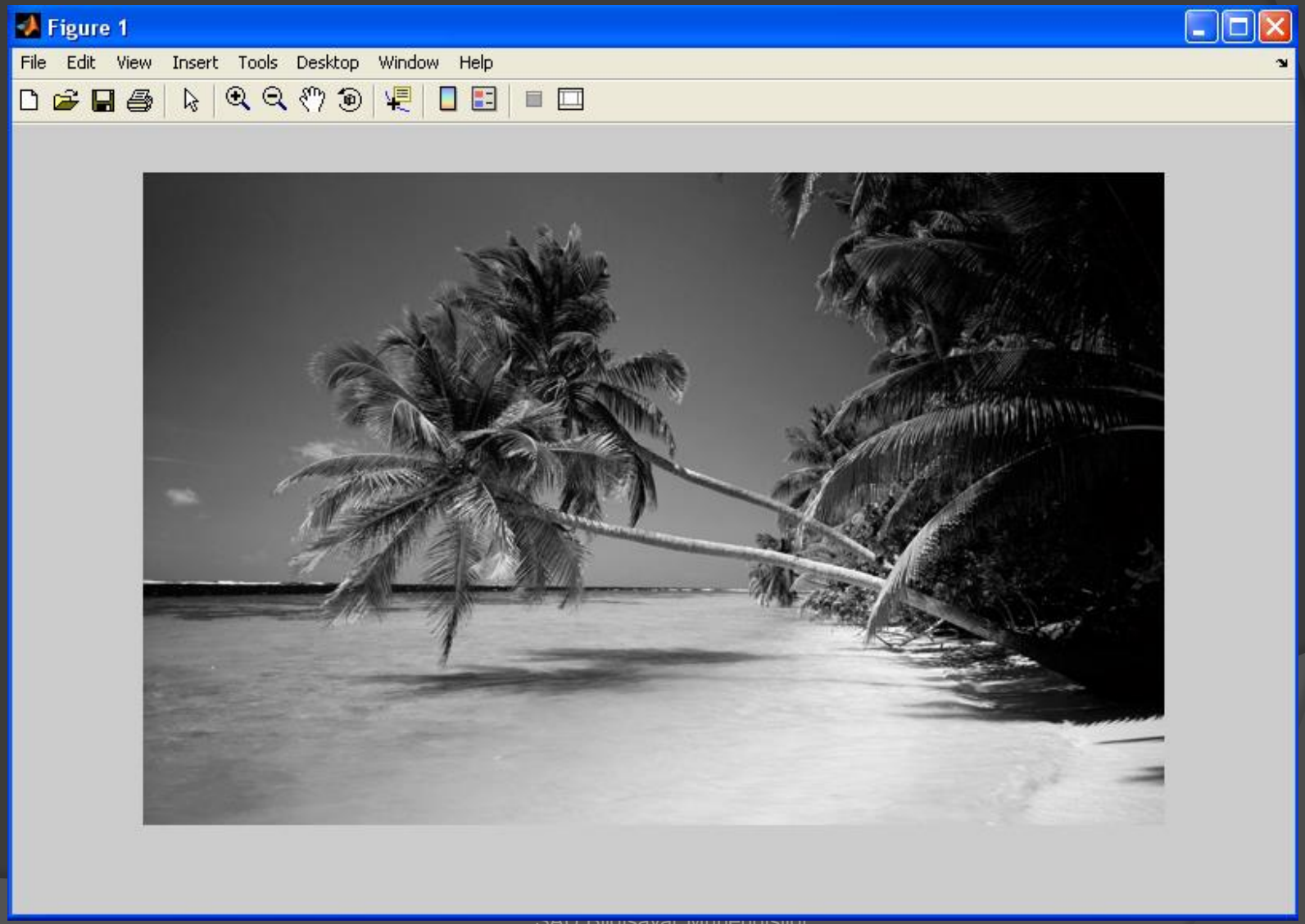


```
I = imread('pout.tif');
J = imadjust(I);
imshow(I), figure, imshow(J)
```

```
RGB1 = imread('football.jpg');
RGB2 = imadjust(RGB1,[.2 .3 0; .6 .7 1],[]);
imshow(RGB1), figure, imshow(RGB2)
```



```
f=imread('c:\r2.jpg');
g=f(:,:,1); %renkli bir görüntünün r bileşeninin gösterilmesi
imshow(g)
```



imhist(g)

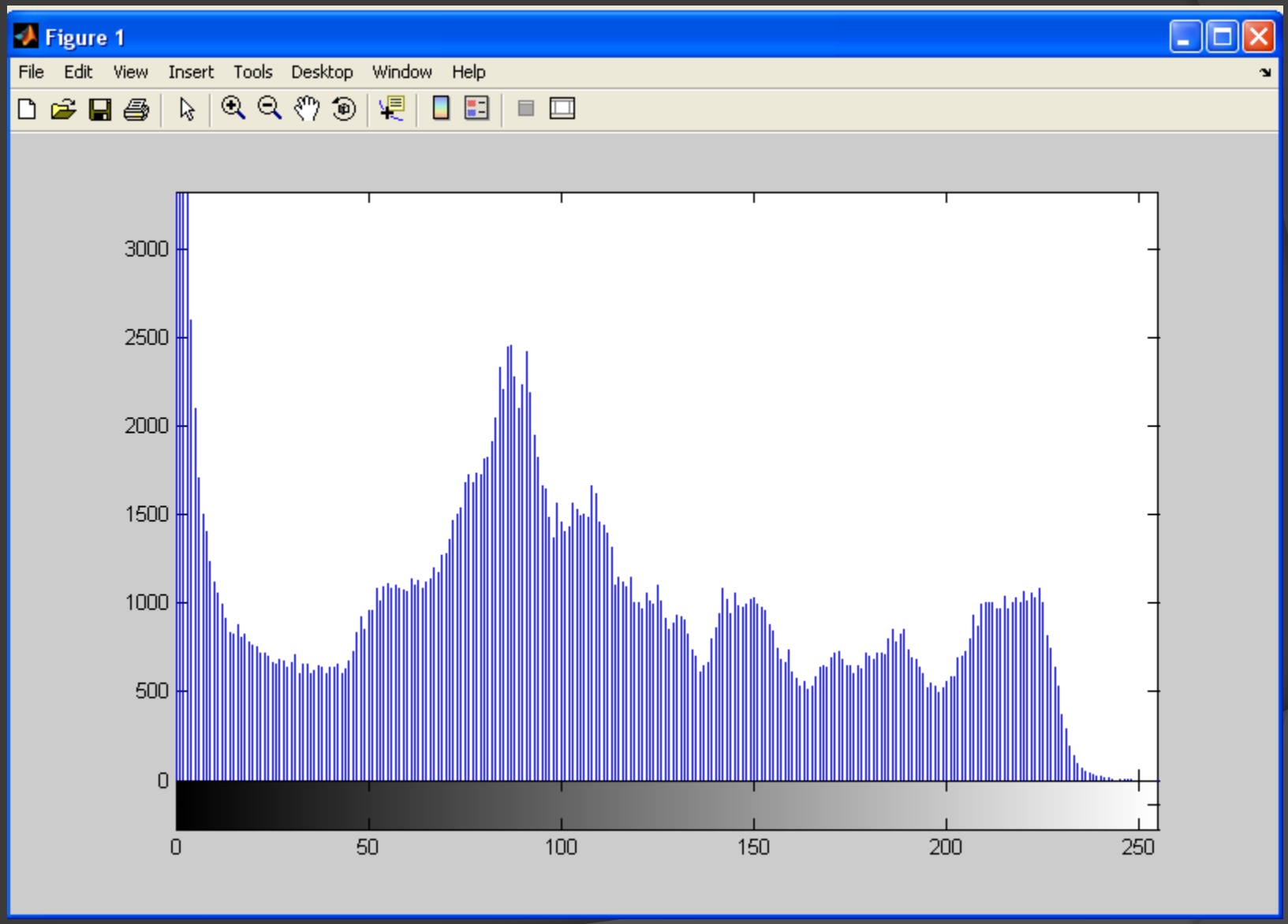
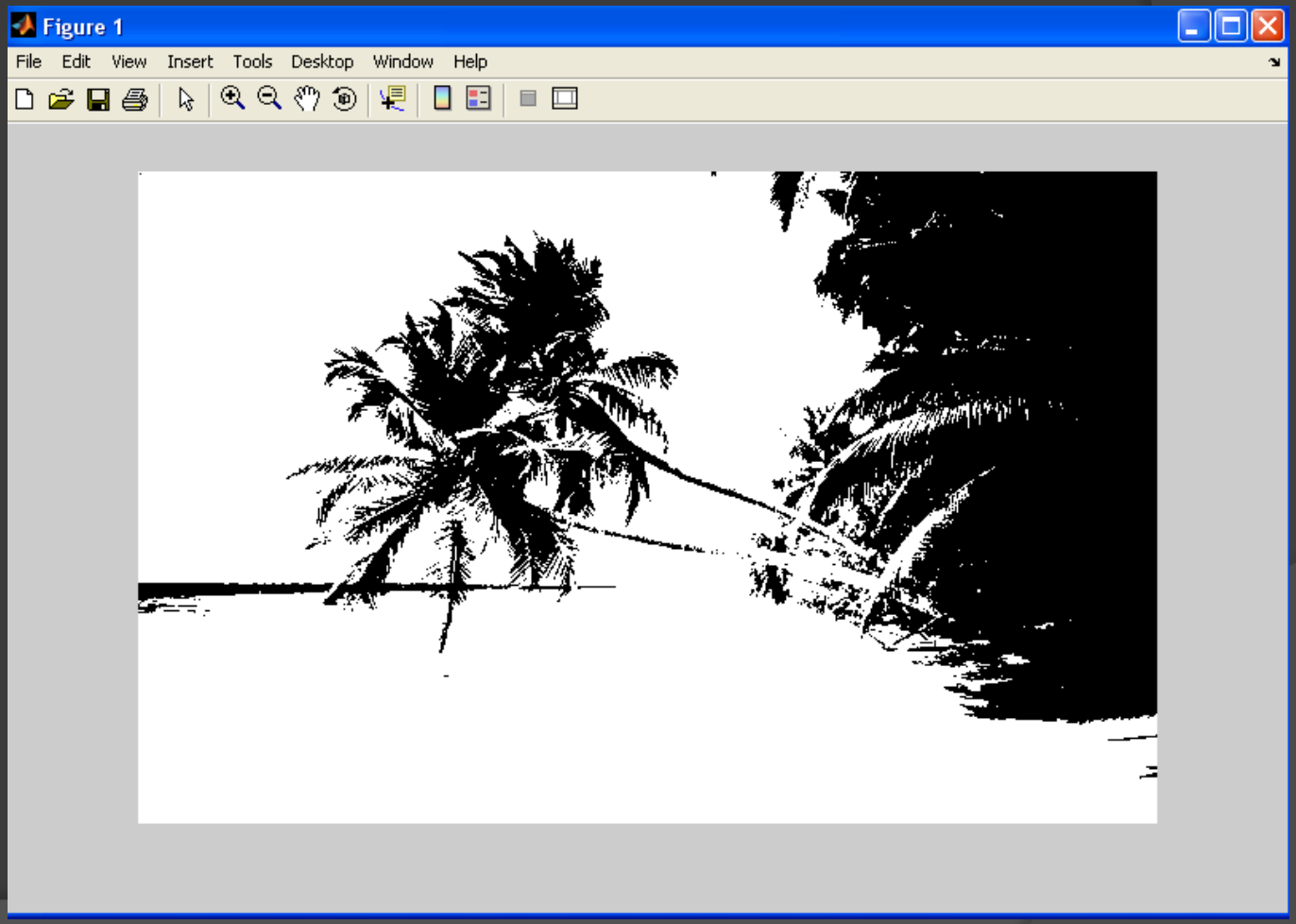


Figure 1
File Edit View Insert Tools Desktop Window Help
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250

```
g=rgb2gray(f);
imshow(g);
figure;imhist(g);
level=graythresh(g)

level =

 0.4275
BW = im2bw(g,level);
figure;imshow(BW);
```



Histogram Denkleştirme

Histogram denkleştirme görüntü işleme yazılımlarının en önemli parçalarından biridir. Bu işlem zıtlığı düzeltir ve histogram denkleştirmenin hedefi tek biçimli bir histogram elde etmektir. Bu teknik bütün görüntü için veya görüntünün bir parçası için kullanılabilir.

Histogram denkleştirme histogramı düzleştirmez. Yoğunluk dağılımını tekrar dağıtır. Eğer bir görüntünün histogramında pek çok tepeler ve vadiler varsa, denkleştirme sonucu yine tepeler ve vadiler olacaktır, fakat tepeler ve vadiler kaydırılır. Bu yüzden histogram denkleştirmeyi tanımlamak için düzleştirmeden ziyade yayma daha iyi bir terim olacaktır.

Histogram denkleştirme bir nokta işlemi olduğu için yeni yoğunluklar görüntüye eklenecektir. Var olan değerler yeni değerlere adreslenir, fakat sonuç görüntüsündeki yoğunlukların sayısı orijinal sayısından daha az veya eşit olacaktır.

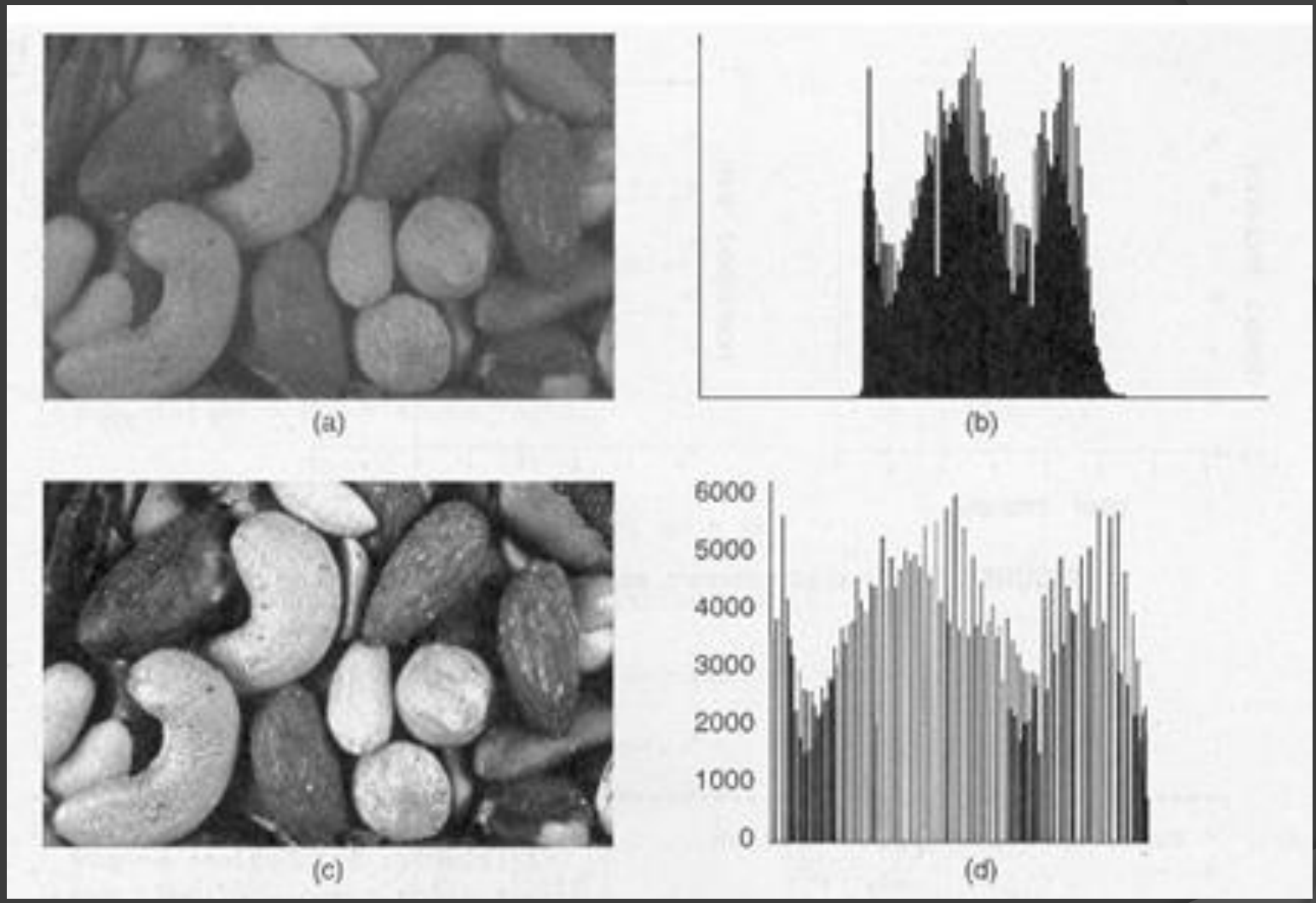
İşlemler

1. Histogramı hesapla
2. Normalleştirilmiş histogramı hesapla
3. Giriş görüntüsünü çıkış görüntüsüne dönüştür.

İlk adım görüntüdeki her piksel değerinin sayılması ile hesaplanabilir. Sıfır dizisi ile başlayabilirsiniz. 8-bitlik pikseller için dizinin boyutu 256'dır (0-255). Görüntüyü ayrıştır ve her bir dizi elemanını uyan piksel değerine göre arttır.
İkinci adım histogram değerlerinin toplamının başka bir dizide saklanmasını gerektirir. Bu dizide I. eleman 0'dan I. elemana kadar olan elemanların toplamını içerecektir. 255. Eleman 0,1,…254,255 histogram elemanlarının toplamını içerecektir. Daha sonra bu dizi her bir elemanı (maksimum piksel değeri)/(piksel sayısı) ile çarpılarak normalleştirilir. 8-bitlik 512x512'lik bir görüntü için bu sabit 255/262144 olacaktır.

Adım 2’nin sonuçlanması için bir LUT(look-up-table) kullanılabilir.

Histogram denkleştirme koyu bölgelerde detayları olan görüntülerde daha iyi çalışır. Bazı kişiler diğer işlemleri uygulamadan önce görüntüye histogram denkleştirme uygular. Bu, iyi bir uygulama değildir çünkü kaliteli görüntülerin kalitesi bozulabilir.



(a) Orijinal görüntü; (b)Orijinal görüntünün histogramı;
(c) Düzeltilmiş görüntü; (d) Düzeltilmiş görüntünün histogramı.

```
eh=histeq(g);
figure;imshow(eh)
figure;imhist(eh);
level=graythresh(eh)

level =

    0.498
bw=im2bw(eh,level);
figure;imshow(bw);
```

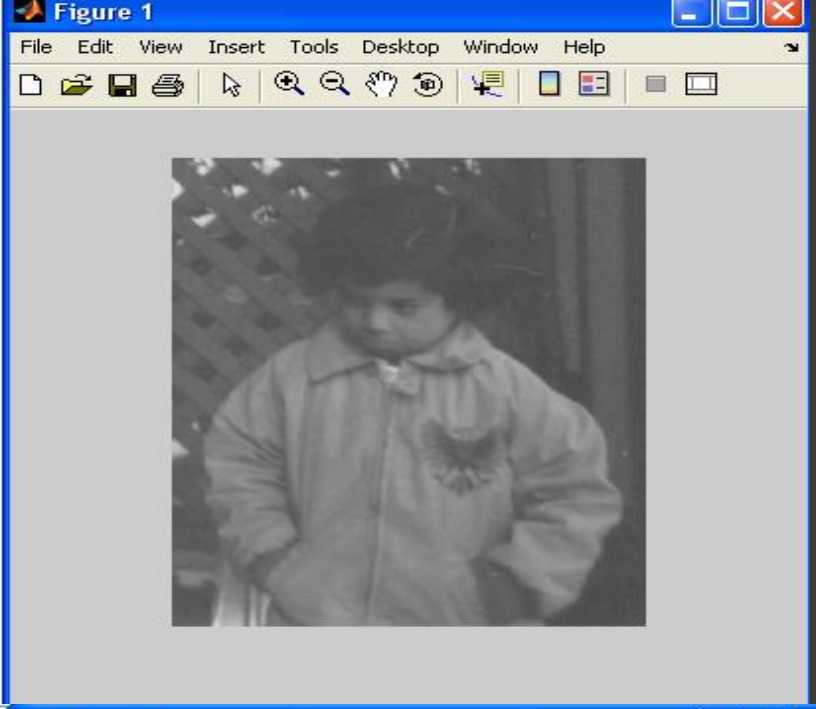


Figure 1
File Edit View Insert Tools Desktop Window Help

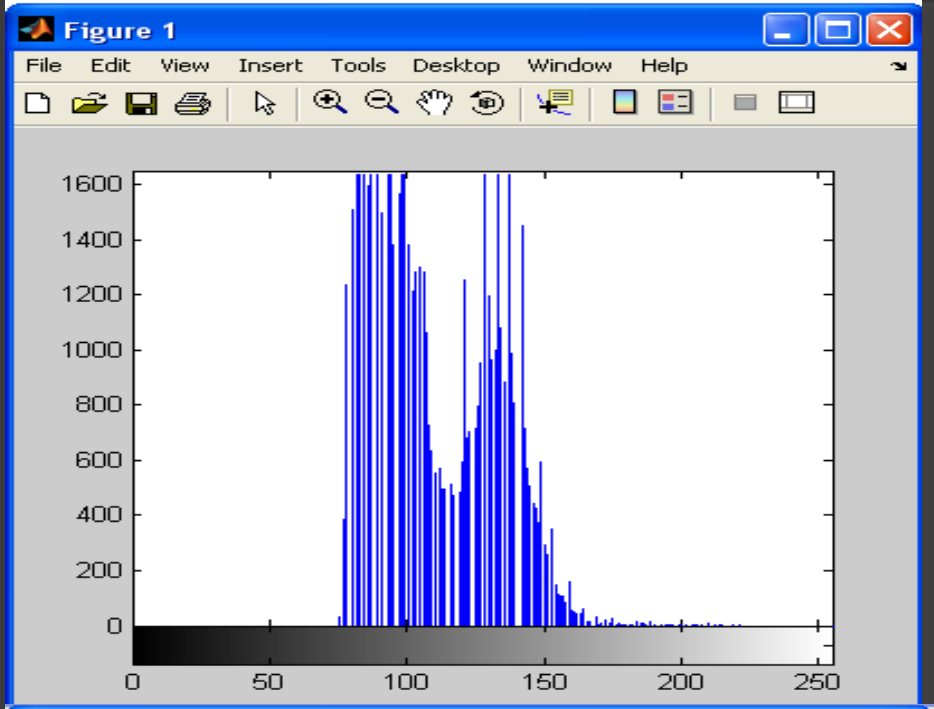


Figure 1
File Edit View Insert Tools Desktop Window Help
1600
1400
1200
1000
800
600
400
200
0
0 50 100 150 200 250

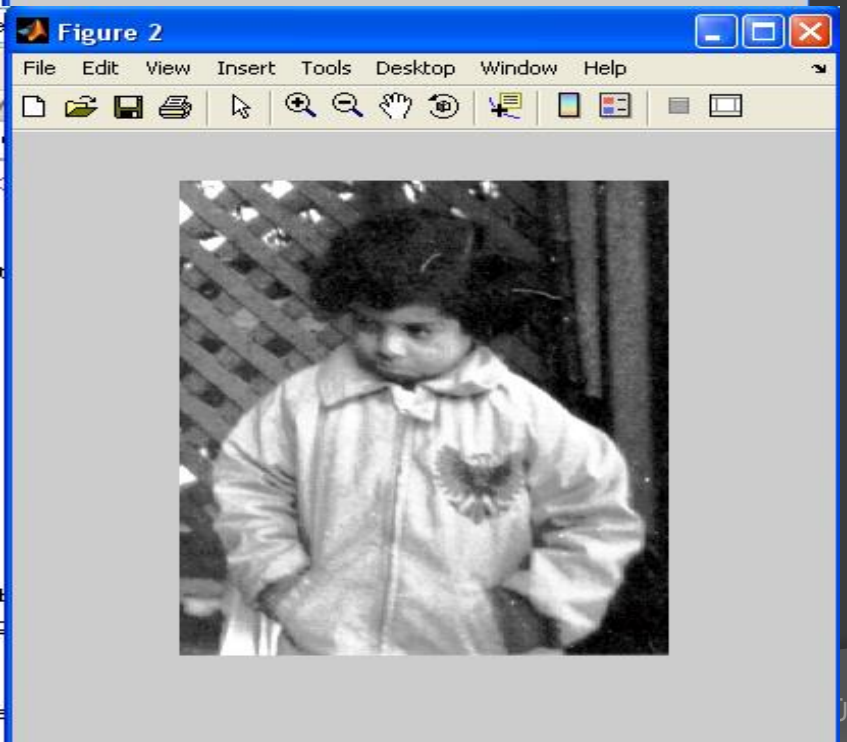


Figure 2
File Edit View Insert Tools Desktop Window Help

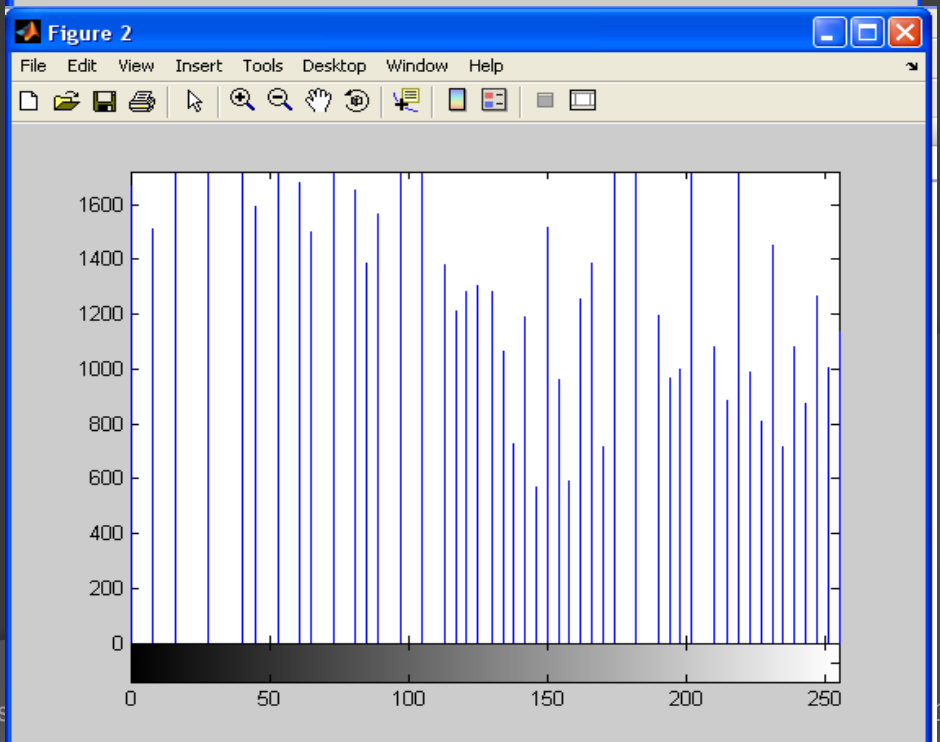


Figure 2
File Edit View Insert Tools Desktop Window Help
1600
1400
1200
1000
800
600
400
200
0
0 50 100 150 200 250

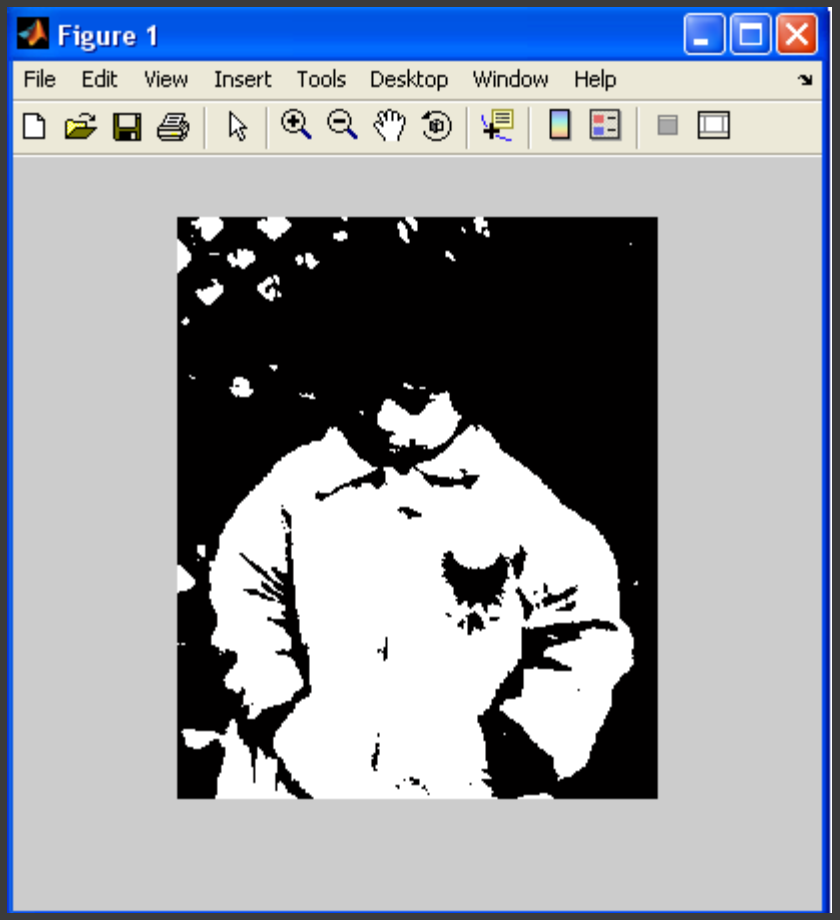


Figure 1
File Edit View Insert Tools Desktop Window Help

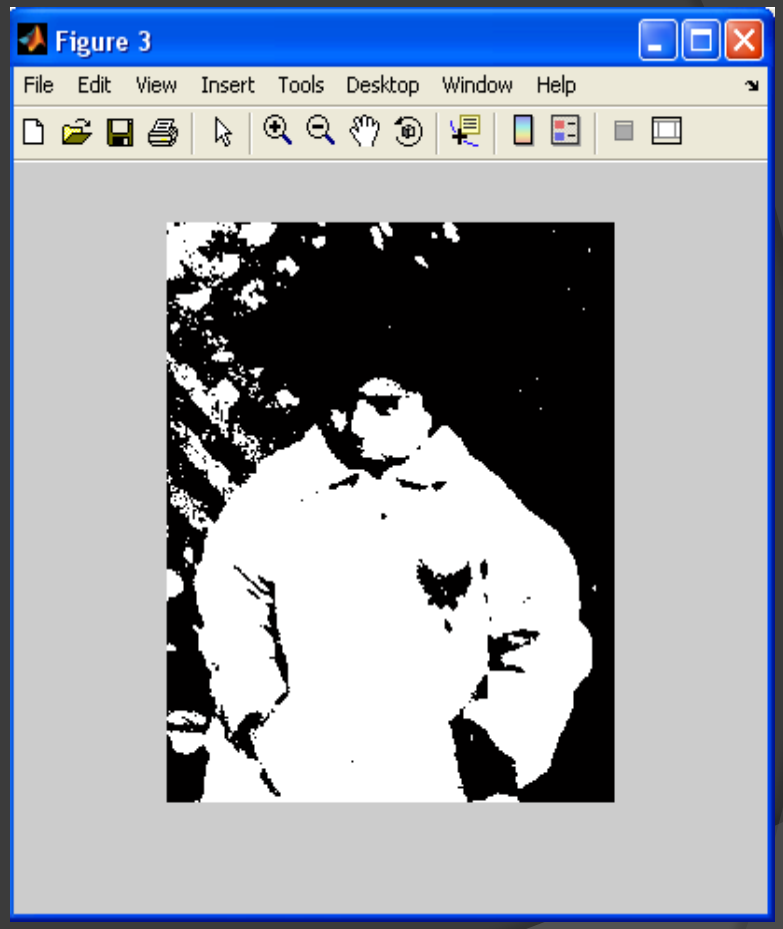


Figure 3
File Edit View Insert Tools Desktop Window Help

Histogram Denkleştirme (Zıtlık Düzeltme)

- Histogramı hesapla
- Kümülatif histogramı hesapla

```
Cum[0]=0;
For(greylevel =1; greylevel < max; greylevel++)
   Cum[greylevel] = Hist[greylevel] + Cum[greylevel-1];
```

- Lookup table kullan

```
for(g=0;g<max;g++)
    Tab[g] = round (double) (max*cum[g])/(rows*cols));
```

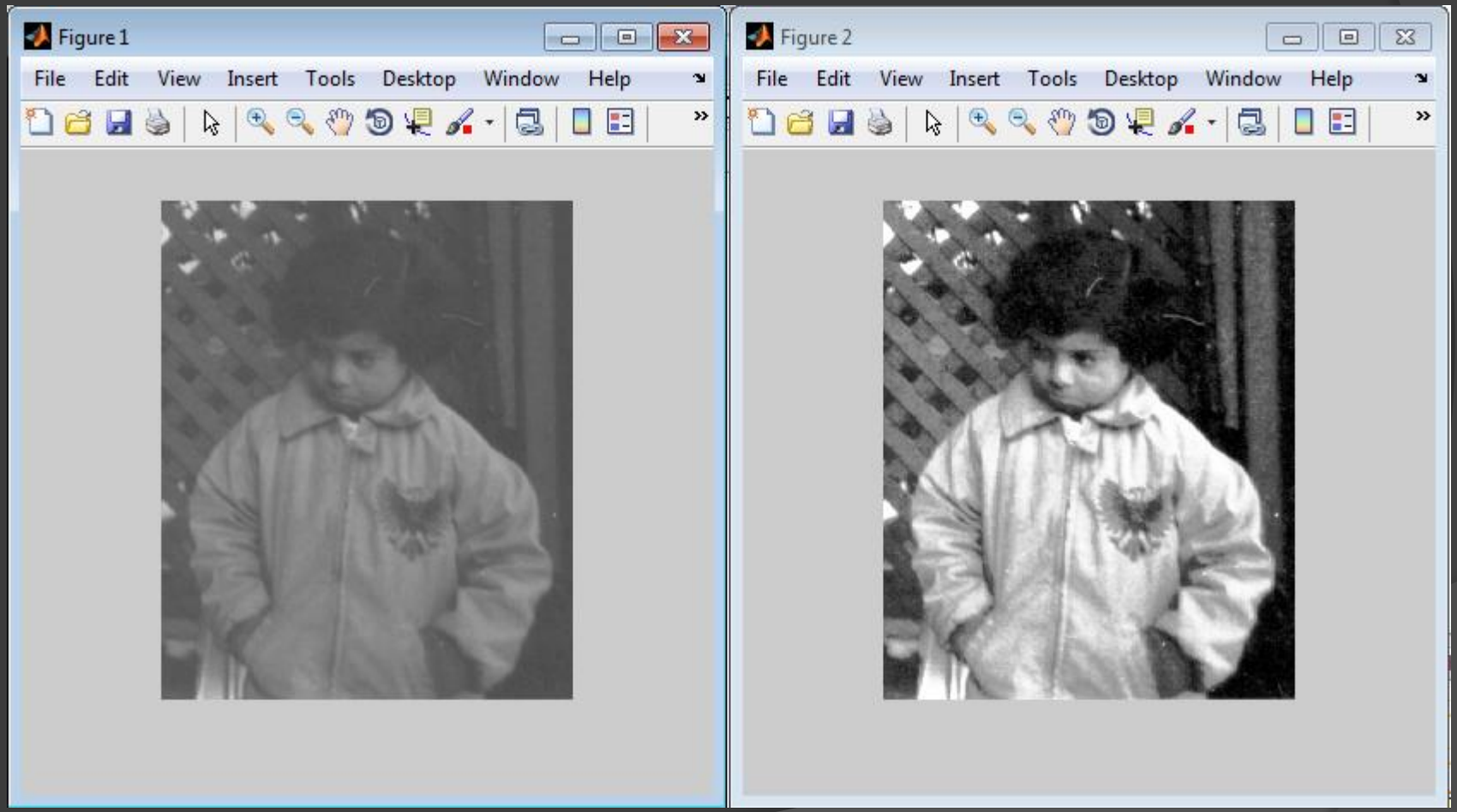
- Yeni grilik seviyelerini hesapla

```
for (r = 1; r< rows; r++) for (c=1;c<cols;c++)
    img[r][c] = Tab[img[r][c]]
```

Histogram Denkleştirme (Zıtlık Düzeltme)

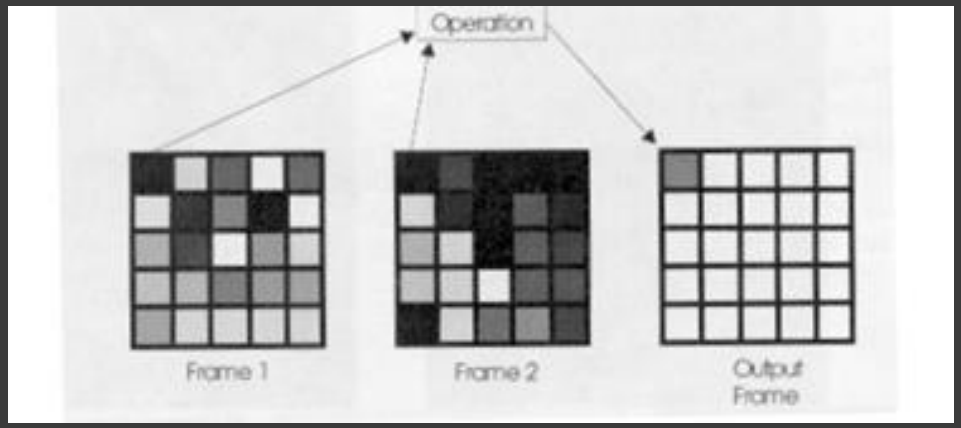
```
1  -    A=imread('pout.tif');
2  -    [r,c]=size(A);
3  -    B=imhist(A);
4  -    cum(1)=0;
5  -    for k=2:256
6  -    cum(k)=B(k)+cum(k-1);
7  -    end
8  -    for g=1:256
9  -    Tab(g)=round(double(256*cum(g)/(r*c)));
10 -    end
11 -    for i=1:r
12 -        for j=1:c
13 -       img(i,j)=Tab(A(i,j));
14 -        end
15 -    end
16 -    imshow(A);
17 -    figure;imshow(uint8(img));
```

Program çıktısı



Çoklu Görüntü İşlemleri

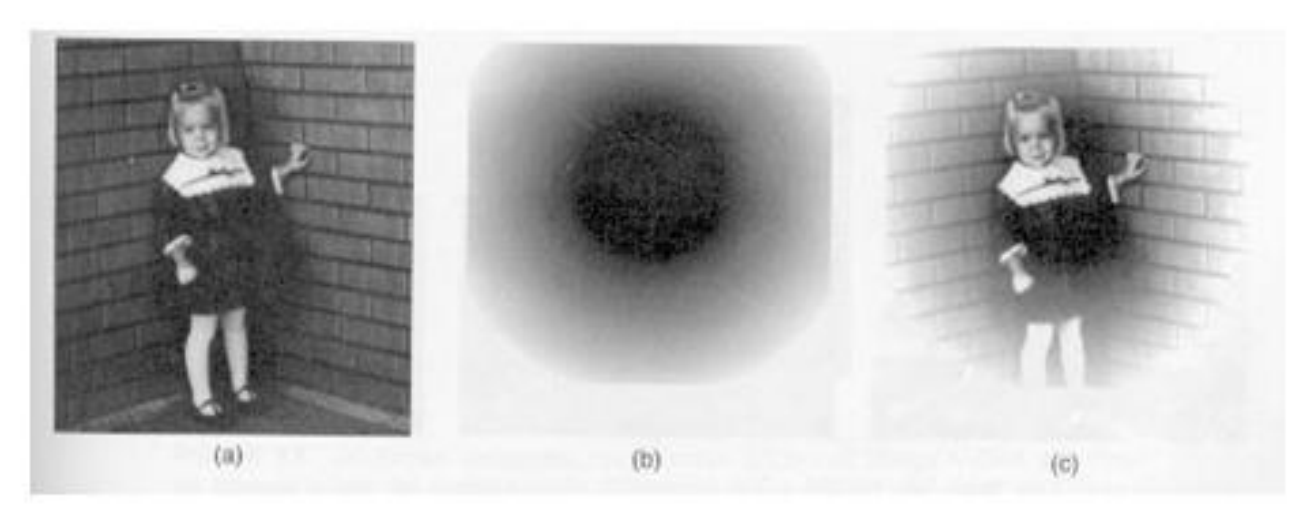
Çerçeve işlemi iki veya daha çok farklı görüntünün işlemine dayanan piksel değerinin oluşturulmasıdır. Burada iki görüntünün pikselleri ile yapılan işlemler ile yeni piksel değerleri oluşturulur. Her bir çıkış pikseli giriş görüntülerinin pikselleri ile aynı pozisyonda yer alır.



Toplama

İlk işlem toplama işlemidir. İki görüntünün birleştirilmesi ile yeni görüntü oluşturulur. Genellikle maksimum değeri aşarak taşmaya neden olacağından direkt olarak toplanmaz. Bir a değeri belirlenir ve toplama işlemi aşağıdaki gibi gerçekleştirilir.

Yeni-piksel=a*piksel1+(1-a)*piksel2



(a) Image 1, (b) Image 2; (c) Image 1 + Image 2.

Böylelikle görüntülerden bir diğerinden daha baskın yapılabilir. Bazı grafik sistemleri her piksel ile ekstra bilgi saklar. Bu bilgi alfa kanalı olarak adlandırılır ve iki görüntünün nasıl birleşeceğini belirler.

Çıkarma

Arka plan çıkarma iki görüntü arasındaki hareketi belirlemek için ve eğer her iki görüntüde de varsa arka plan gölgelenmesini çıkarmak için kullanılabilir. Görüntüler herhangi bir aydınlatma durumuna bakılmaksızın herhangi bir anda çekilmiş olabilir. Eğer nesne arka plandan daha aydınlık bir hale dönüşmüş ise tam tersi yapılır.
Çıkarma, bir görüntüdeki her bir pikselin grilik seviyesinden diğer görüntünün aynı pikselinin grilik seviyesini çıkarmak anlamına gelir. x>y olmak üzere,

Sonuç=x-y

Eğer x<y ise sonuç negatif olacaktır. Eğer değerler unsigned karakterleri tutuyorsa en yüksek pozitif değere dönüştürülür. Örneğin

-1 255 değerini alır
-2 254 değerini alır

Daha iyi bir işlem için

Sonuç=|x-y| kullanılır.

Burada nesnenin arka plandan açık ya da koyu olup olmadığına bakılmaz. Bu nesnenin negatif görüntüsünü verecektir. Bunu pozitife dönüştürmek için elde edilen grilik seviyesi maksimum grilik seviyesinden çıkarılır.

Yeni-görüntü=MAX-|x-y|

```
ColorRGB image1[320][240];
ColorRGB image2[320][240];
ColorRGB image3[320][240];
ColorRGB result[320][240];

int main(int argc, char *argv[])
{
    //set up the screen
    screen(320,240,0, "Image Arithmetic");

    //load the images into the buffers
    loadBMP("pics/photo1.bmp", image1[0], w, h);
    loadBMP("pics/photo2.bmp", image2[0], w, h);
    loadBMP("pics/photo3.bmp", image3[0], w, h);

    //do the image arithmetic (here: "average")
    for(int x = 0; x < w; x++)
    for(int y = 0; y < h; y++)
    {
        result[x][y].r = (image1[x][y].r + image2[x][y].r) / 2;
        result[x][y].g = (image1[x][y].g + image2[x][y].g) / 2;
        result[x][y].b = (image1[x][y].b + image2[x][y].b) / 2;
    }

    //draw the result buffer to the screen
    for(int x = 0; x < w; x++)
    for(int y = 0; y < h; y++)
    {
        pset(x, y, result[x][y]);
    }

    //redraw & sleep
    redraw();
    sleep();
}
```



Toplama

Toplama görüntülerin her birindeki ilgili renk kanallarının toplanmasıdır. Her bir renk bileşeni 0 ve 255 arasında değişir. Eğer iki rengin toplamı 255'den büyük olursa 255 ile sınırlandırılır. Aşağıdaki kodlamayı döngünün içine kopyalayıp yapıştırarak iki resmin toplamının sonucunu görebilirsiniz.

```
result[x][y].r = min(image2[x][y].r + image3[x][y].r, 255);
result[x][y].g = min(image2[x][y].g + image3[x][y].g, 255);
result[x][y].b = min(image2[x][y].b + image3[x][y].b, 255);
```

Çıkarma

Çıkarma işlemi de benzer şekilde yapılır, fakat negatif değerler 0 ile sınırlandırılır.

```
result[x][y].r = max(image2[x][y].r - image1[x][y].r, 0);
result[x][y].g = max(image2[x][y].g - image1[x][y].g, 0);
result[x][y].b = max(image2[x][y].b - image1[x][y].b, 0);
```

İnekli resim atlı resimden çıkarılmıştır. Dolayısı ile inekler negatife dönüştürülmüştür. Bu, morumsu renklerin oluşmasına neden olmuştur. Parlak gökyüzünün çiçeklerden çıkarılması yukarı kısmın siyaha dönüşmesine neden olmuştur.

Çıkarmanın sırası önemlidir. İkinci resim üçüncü resimden çıkarıldığında aşağıdaki gibi sonuçlanır.

Fark

Fark, çıkarma ile hemen hemen aynıdır. Sadece negatif değerlerin mutlak değeri alınarak sınırlandırılır. Dolayısı ile her iki görüntünün renkleri arasındaki farkı elde ederiz.

```
result[x][y].r = abs(image1[x][y].r - image2[x][y].r);
result[x][y].g = abs(image1[x][y].g - image2[x][y].g);
result[x][y].b = abs(image1[x][y].b - image2[x][y].b);
```

Çarpma

Çarpma işlemi için 0'dan 255'e kadar olan renk bileşenleri çarpılmaz. Bunu yaparsak maksimum değer 65025 olur. Bunun yerine değerler 0 ve 1 arasındaki floating point değerlerine dönüştürülür. Ve bunlar çarpılır. Böylece sonuç daima 0 ve 1 arasında olur. Çarpma sonucu 255 ile çarpılır.

```
result[x][y].r = int(255 * (image2[x][y].r / 255.0 * image1[x][y].r / 255.0));
result[x][y].g = int(255 * (image2[x][y].g / 255.0 * image1[x][y].g / 255.0));
result[x][y].b = int(255 * (image2[x][y].b / 255.0 * image1[x][y].b / 255.0));
```

İnekli görüntüdeki parlak gökyüzü 1 değerinde görülebilir ve bunun atlı resimdeki 1'ler ile çarpılması sonucu resmin bazı bölümlerinin atlı resimdeki ile aynı olduğu görülebilir. İki resmin de alt kısımları daha koyu olduğundan sonuç görüntüsünün alt kısmı daha koyudur.

Ortalama

Ortalama iki görüntünün toplanıp sonucun 2'ye bölünmesi ile hesaplanır.

```
result[x][y].r = (image1[x][y].r + image2[x][y].r) / 2;
result[x][y].g = (image1[x][y].g + image2[x][y].g) / 2;
result[x][y].b = (image1[x][y].b + image2[x][y].b) / 2;
```

Geçiş Solgunluğu

Geçiş solgunluğu ağırlıklı ortalama kullanılarak yapılır. İlk önce ilk görüntüye yüksek ağırlık ve ikinciye düşük ağırlık verilir. Fakat zamanla ikinci görüntünün ağırlığı arttırılır ve birinci görüntününki iyi bir gölge ile sonuçlanıncaya kadar azaltılır. Aşağıdaki örnekte birinci görüntünün ağırlık faktörü 0.75 ve ikinci görüntününki 0.25 olarak ayarlanmıştır.

```
result[x][y].r = int(image1[x][y].r * 0.75 + image2[x][y].r * 0.25);
result[x][y].g = int(image1[x][y].g * 0.75 + image2[x][y].g * 0.25);
result[x][y].b = int(image1[x][y].b * 0.75 + image2[x][y].b * 0.25);
```

Minimum ve maksimum

Bu işlem piksel değerinin en yükseği veya en düşüğünün alınması ile yapılır. Örneğin ikisinden minimum olanın alınması aşağıdaki gibi olur.

```
result[x][y].r = min(image1[x][y].r, image2[x][y].r);
result[x][y].g = min(image1[x][y].g, image2[x][y].g);
result[x][y].b = min(image1[x][y].b, image2[x][y].b);
```

```
result[x][y].r = max(image1[x][y].r, image2[x][y].r);
result[x][y].g = max(image1[x][y].g, image2[x][y].g);
result[x][y].b = max(image1[x][y].b, image2[x][y].b);
```

Genlik

```
result[x][y].r = int(sqrt(double(image1[x][y].r * image1[x][y].r + image2[x][y].r * image2[x][y].r)) / sqrt(2.0));
result[x][y].g = int(sqrt(double(image1[x][y].g * image1[x][y].g + image2[x][y].g * image2[x][y].g)) / sqrt(2.0));
result[x][y].b = int(sqrt(double(image1[x][y].b * image1[x][y].b + image2[x][y].b * image2[x][y].b)) / sqrt(2.0));
```

AND, OR ve XOR

```
result[x][y].r = image1[x][y].r & image2[x][y].r;
result[x][y].g = image1[x][y].g & image2[x][y].g;
result[x][y].b = image1[x][y].b & image2[x][y].b;
```

Or would you prefer OR?

```
result[x][y].r = image1[x][y].r | image2[x][y].r;
result[x][y].g = image1[x][y].g | image2[x][y].g;
result[x][y].b = image1[x][y].b | image2[x][y].b;
```

```
result[x][y].r = image1[x][y].r ^ image2[x][y].r;
result[x][y].g = image1[x][y].g ^ image2[x][y].g;
result[x][y].b = image1[x][y].b ^ image2[x][y].b;
```

Birkaç Matlab Fonksiyonu

J= imabsdiff(I, K); (çıkarma)
J= imadd(I, K); (toplama)
J= imcomplement(I); (tersini alma)
J= imdivide(I, background); (bölme)
J= immultiply(I, 1.2); (çarpma)
J= imsubtract(I, K); (çıkarma)
J= imresize(I, 1.25); (boyut değiştirme)
J= imrotate(I, 35, 'bilinear'); (döndürme)
I= imcrop; (kesme)