Bilgisayar Görmesi

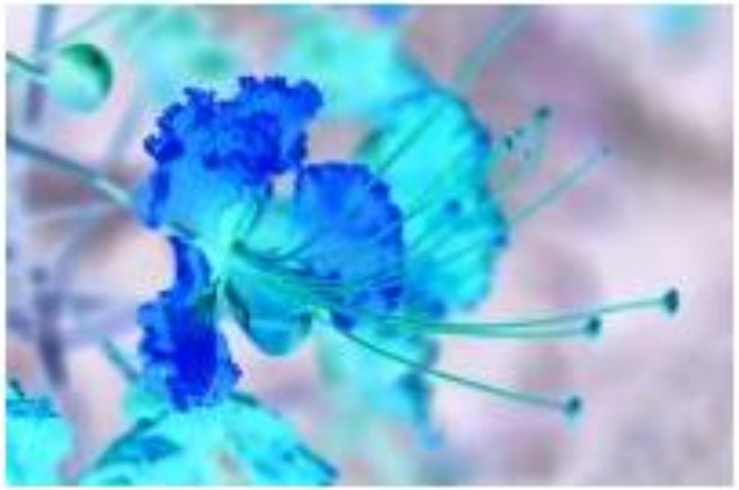
Ders 5:FİLTRELER

Dr. Öğr. Üyesi Serap ÇAKAR

Negatif Görüntüler

Aşağıdaki kodlar 200x133'lük bir BMP görüntüsünü yükler, negatifini hesaplar ve sonucu gösterir. ColorRGB kodu rgb renklerini tanımlayan r, g ve b tamsayı değerlerini içeren yapıdır.

```
ColorRGB image[200][133];
int main(int argc, char *argv[])
{
    screen(200, 133, 0, "RGB Color");
    loadBMP("pics/flower.bmp",image[0], 200, 133);

    ColorRGB color; //the color for the pixels

    for(int x = 0; x < w; x++)
    for(int y = 0; y < h; y++)
    {
        //here the negative color is calculated!
        color.r = 255 - image[x][y].r;
        color.g = 255 - image[x][y].g;
        color.b = 255 - image[x][y].b;
        pset(x, y, color);
    }

    redraw();
    sleep();
    return 0;
}
```



Parlaklığın Değiştirilmesi

Parlaklığı Değiştirmek için R, G ve B değerlerini 1'den büyük bir rakam ile bölmek görüntüyü koyulaştırır. Eğer renk bileşenleri 255'den daha büyük olursa 255 ile sınırlandırılır.

Örneğin görüntüyü iki kat koyulaştırmak istiyorsak bir önceki slayttaki negatife dönüştüren 3 satır aşağıdaki 3 satır ile değiştirilir.

```
color.r = image[x][y].r / 2;
color.g = image[x][y].g / 2;
color.b = image[x][y].b / 2;
```

Veya 1.5 kat koyulaştırmak için;

Or to make it 1.5 times as dark, use:

```
color.r = int(image[x][y].r / 1.5);
color.g = int(image[x][y].g / 1.5);
color.b = int(image[x][y].b / 1.5);
```

2 kat parlaklaştırmak için;

To make it twice as bright, use:

```
color.r = image[x][y].r * 2;
color.g = image[x][y].g * 2;
color.b = image[x][y].b * 2;

if(color.r > 255) color.r = 255;
if(color.g > 255) color.g = 255;
if(color.b > 255) color.b = 255;
```

```
color.r = image[x][y].r + 50;
color.g = image[x][y].g + 50;
color.b = image[x][y].b + 50;

if(color.r > 255) color.r = 255;
if(color.g > 255) color.g = 255;
if(color.b > 255) color.b = 255;
```

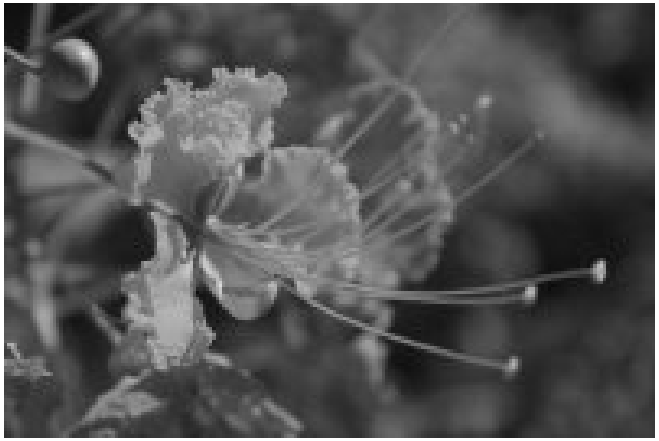
Or darker:

```
ccolor.r = image[x][y].r - 50;
color.g = image[x][y].g - 50;
color.b = image[x][y].b - 50;

if(color.r < 0) color.r = 0;
if(color.g < 0) color.g = 0;
if(color.b < 0) color.b = 0;
```

Gri Seviye

Gri seviyeli görüntüyü hesaplama yollarından biri üç renk bileşeninin ortalamasını alarak grilik değeri olarak kullanmaktır.

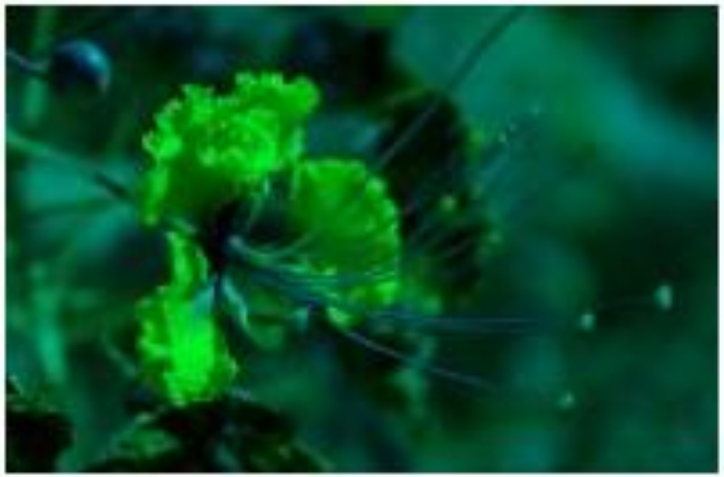
```
color.r = image[x][y].r - 50;
color.g = image[x][y].g - 50;
color.b = image[x][y].b - 50;
color.r = color.g = color.b = (color.r + color.g + color.b) / 3;
```



Kanalları Değiştirme ve Çıkartma

Kanalların çıkarılması ile bir görüntünün bir renk bileşenini çıkartırız, örneğin eğer çiçek görüntüsünden kırmızı bileşeni çıkartırsanız görüntü aşağıdaki gibi olur. Yeşil bileşeni çıkarırsak sağ üst, mavi bileşeni çıkarırsak sağ alttaki gibi olur.

```
color.r = 0; //red component set to zero
color.g = image[x][y].g;
color.b = image[x][y].b;
```



If you remove green instead, you get:

And if you remove blue, you get:

İki kanalın yerinin değiştirilmesi genel olarak yeni renklerin oluşmasına neden olur, örneğin kırmızı ve yeşil bileşenlerin yeri değiştirildiğinde arka plan kırmızımsı olurken çiçek yeşile dönüşür. Mavi ve kırmızı yer değiştirdiğinde çiçek maviye dönüşür.

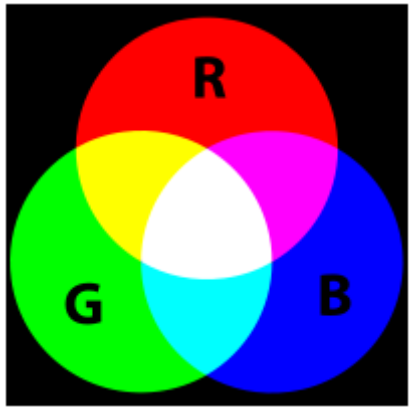
```
color.r = image[x][y].g; //the green component of the image
color.g = image[x][y].r; //the red component of the image
color.b = image[x][y].b; //the blue component of the image
```

And if red and blue are swapped instead, the flower becomes of course blue:

Çiçeği sarıya dönüştürmek için R ve G bileşenleri kırmızıya dönüştürülür.

To make the flower yellow, set both R and G to the red channel of the image:

```
color.r = image[x][y].r; //the red component of the image
color.g = image[x][y].r; //the red component of the image
color.b = image[x][y].b; //the blue component of the image
```



http://student.kuleuven.be/~m0216922/CG/color.html#RGB_Arithmetic_

Matlab’de kanalları değistirme ve çıkarma

```
F1=imread('C:\flower.jpg');
imshow(F1);
G1=F1(:,:,1);
G2=F1(:,:,2);
G3=F1(:,:,3);
G4=255-G1;
G5=255-G2;
G6=255-G3;
F2(:,:,1)=G4;
F2(:,:,2)=G5;
F2(:,:,3)=G6;
figure;imshow(F2);
```

```
F3(:,:,1)=0;
F3(:,:,2)=G2;
F3(:,:,3)=G3;
figure;imshow(F3);

F3(:,:,1)=G1;
F3(:,:,2)=0;
F3(:,:,3)=G3;
figure;imshow(F3);
```

```
F3(:,:,1)=G1;
F3(:,:,2)=G2;
F3(:,:,3)=0;
figure;imshow(F3);
```

```
F3(:,:,1)=G2;
F3(:,:,2)=G1;
F3(:,:,3)=G3;
figure;imshow(F3);
```

```
F3(:,:,1)=G3;
F3(:,:,2)=G2;
F3(:,:,3)=G1;
figure;imshow(F3);
```

```
F3(:,:,1)=G1;
F3(:,:,2)=G1;
F3(:,:,3)=G3;
figure;imshow(F3);
```

Görüntü Filtreleme

Görüntü filtreleme, gürültü çıkartma ve görüntü iyileştirme gibi pencere işlemleri uygulamalarını içerir. Bu bölüm yerel ortalama, medyan veya mod filtreleri gibi temel uygulamalar ile ne yapılabileceği ile ilgilidir. Genellikle gri seviyeli görüntüler üzerinde çalışılmıştır.

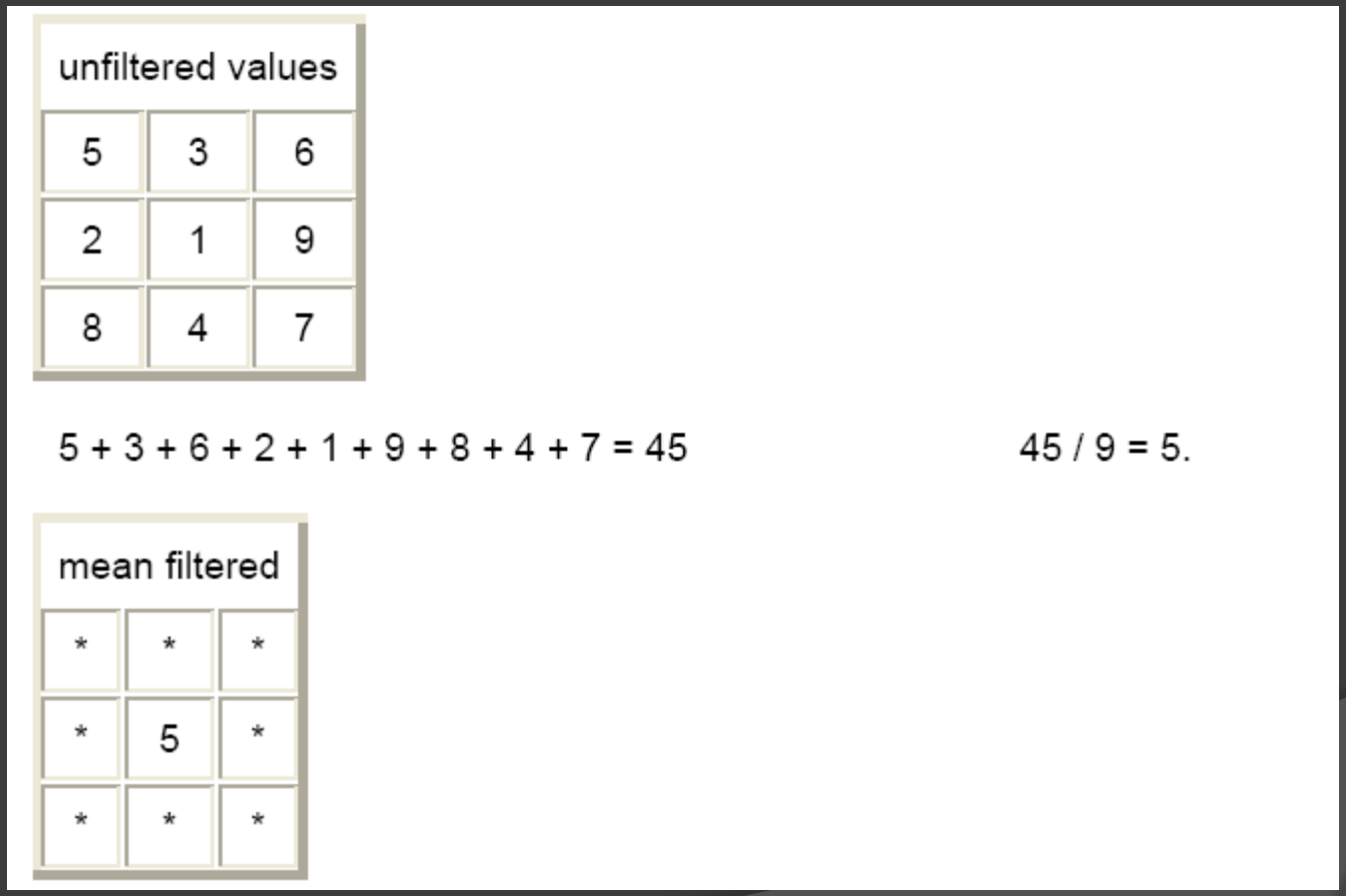
Düzleştirme ile Gürültü Bastırma

Alçak geçiren filtre normalde yüksek frekanslı işaret bileşenlerinin elimine edilmesi olarak düşünülür. Bu yüzden özel frekans uzayında gerçekleştirilir. Yine de bunu piksel uzayında gerçekleştirmek mümkündür.

1966'da Rosie'nin yapmış olduğu çalışmaya dayanarak bir işareti özel bir frekans uzayında bir fonksiyon ile çarpmak fonksiyonun piksel uzayında Fourier dönüşümü ile konvolüsyonunu almaya eşit olduğu söylenebilir.

Mean Filtresi (Ortalama Filtresi)

Mean filtresi penceredeki bütün piksel değerlerinin ortalamasının ortadaki değere yerleştirilmesi ile yapılan özel bir filtredir. Pencere genellikle karedir, herhangi bir şekilde de olabilir. 3x3'lük pencere ile yapılan mean filtresinin bir örneği aşağıda gösterilmiştir.



Şekil: Merkezdeki 1 değeri dokuz değerin ortalaması olan 5 ile değiştirilmiştir.

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Mean Filtresi

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Source: S. Seitz

Orijinal görüntü

3x3 mean filtre uygulanmış

7x7 mean filtre uygulanmış

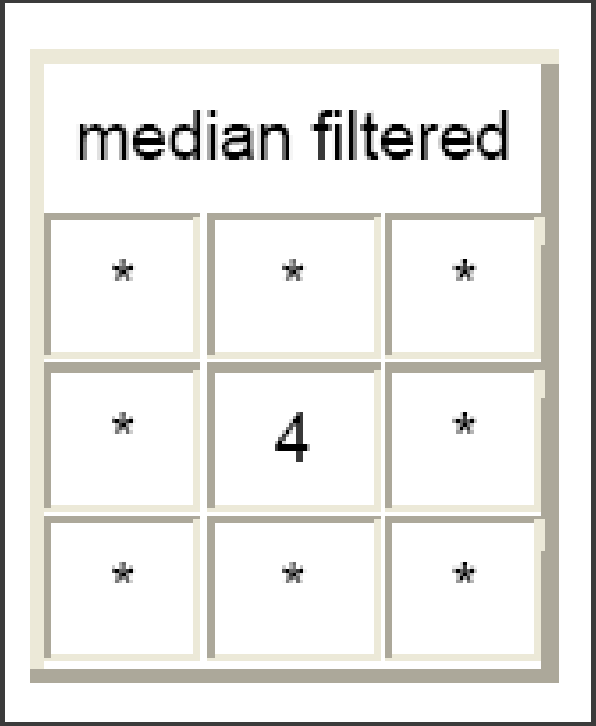
Üç kere 3x3 mean filtre uygulanmış

Medyan Filtresi

Medyan filtresi de özel bir filtredir fakat penceredeki bütün piksel değerlerini küçükten büyüğe doğru sıraladığımızda ortadaki değeri alarak pencerenin ortasına yerleştiririz. Çekirdek çoğu zaman karedir fakat herhangi bir şekilde de olabilir. Aşağıda 3x3'lük pencere kullanılarak yapılan medyan filtreleme örneğinin değerleri görülmektedir.

unfiltered values		
6	2	0
3	97	4
19	3	10

in order: 0, 2, 3, 3, **4**, 6, 10, 15, 97



Merkezdeki değer (daha önce 97 olan) 9 değerin ortasındaki değer (4) ile yer değiştirir.

Birinci örnekte sıralanmış değerler 1, 2, 3, 4, 5, 6, 7, 8, 9 olduğundan medyan filtresi yine 5 değeri ile geri döner. İkinci örnek için mean filtresi 9 değerin ortalaması 114/9=16 olduğundan 16 değeri ile geri döner. Bu, medyan filtresinin iyi bir özelliğini gösterir: bu yetenek darbe gürültüsünü kaldırma özelliğidir (çok yüksek ve çok küçük değerler de dahil). Medyan filtresi bulanıklaştırma yapmadan kenarları koruma özelliğine de sahiptir. Bununla birlikte gürültü olduğunda kenarları biraz bulanıklaştırabilir.

Bozulmuş görüntü

3x3 medyan filtresi uygulanmış

7x7 medyan filtresi uygulanmış

Üç kere 3x3 medyan filtresi uygulanmış

Aşağıdaki örnek ortalama filtresi ve medyan filtresi kullanılarak tuz biber gürültüsünün nasıl kaldırıldığını (Matlab kodları ile) göstermektedir. Bu tip gürültü piksel değerlerini rastgele olarak siyah ve beyaza dönüştürür. Her iki durumda da filtre için 3x3 komşuluğu kullanılmıştır.

1. Görüntüyü oku ve göster

```
I = imread('eight.tif');
imshow(I)
```

2. Gürültü ekle

```
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(J);
```

3. Gürültülü görüntüyü filtrele

```
K = filter2(fspecial('average',3),J)/255;
figure, imshow(K)
```

4.Gürültülü görüntüyü fitrelemek için medyan filtresi kullan ve sonucu göster. Medfilt2 fonksiyonunun gürültüyü kaldırmakta daha iyi sonuç verdiğine ve kenarları daha az bulanıklaştırdığına dikkat et.

```
L = medfilt2(J,[3 3]);
figure, imshow(K)
figure, imshow(L)
```

Konvolüsyon

Konvolüsyon görüntü işleme kullanıcıları için çok yaygın olan temel ve basit bir matematiksel işlemdir. Konvolüsyon genellikle farklı boyutlarda olan iki sayı dizisinin birbiri ile çarpılmasının bir yoludur. Bu, çıkış piksel değerleri giriş piksel değerlerinin basit lineer bir kombinasyonu olan işlemleri yerine getirmek için görüntü işlemede kullanılır.

Konvolüsyon

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$$

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l)$$

Gauss Düzleştirmesi

Aşağıdaki şekil bir Gauss konvolüsyon çekirdeğinin tam sayı değerlerini gösterir.

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Bozulmuş görüntü

Gauss ile düzleştirilmiş

Bulanıklaştırma

Bulanıklaştırma örneğin geçerli piksel ve 4 komşusunun ortalamasının alınması ile yapılır. Geçerli piksel ve 4 komşusunun toplamı alınarak 5'e bölünür veya filtrede beş kere 0.2 kullanılabilir:

```
double filter[filterWidth][filterHeight] =
            { 0.0, 0.2, 0.0,
              0.2, 0.2, 0.2,
              0.0, 0.2, 0.0 };
```

Veya, double factor = 1.0/5;
double bias = 0.0;

Böyle küçük filtre matrisleri ile çok yumuşak bir bulanıklaştırma sağlanır:

Orijinal görüntü

bulanıklaştırılmış görüntü

Daha büyük bir filtre ile resmi daha fazla bulanıklaştırabilirsiniz (filterWidth ve filterHeight değerlerini değiştirmeyi unutmayın):

```
double filter[filterWidth][filterHeight] =
            { 0, 0, 1, 0, 0,
              0, 1, 1, 1, 0,
              1, 1, 1, 1, 1,
              0, 1, 1, 1, 0,
              0, 0, 1, 0, 0,
               };

double factor = 1.0 / 13.0;
double bias = 0.0;
```

Filtre elemanlarının toplamının 1 olması gerekir, fakat filtre içine floating point değerleri doldurmak yerine factor elemanlarının toplamı olan 13'e bölünür.

Orijinal görüntü

bulanıklaştırılmış görüntü

Daha fazla bulanıklaştırmak için aynı küçük filtre birkaç defa da uygulanabilir.

Hareket Bulanıklığı

Hareket bulanıklığı sadece 1 yöne bulanıklaştırma yapılarak gerçekleştirilir. Aşağıda 9x9'luk hareket bulanıklık filtresi görülmektedir.

```
double filter[filterWidth][filterHeight] =
{
    1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 1, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 1, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 1, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 1, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1,
};

double factor = 1.0 / 9.0;
double bias = 0.0;
```

Kenar Bulma

Aşağıda bütün yönlerde kenarları tespit eden basit bir kenar tespit filtresi görülmektedir.

```
double filter[filterWidth][filterHeight] =
{
    -1, -1, -1,
    -1,  8, -1,
    -1, -1, -1
};

double factor = 1.0;
double bias = 0.0;
```

Keskinleştirme

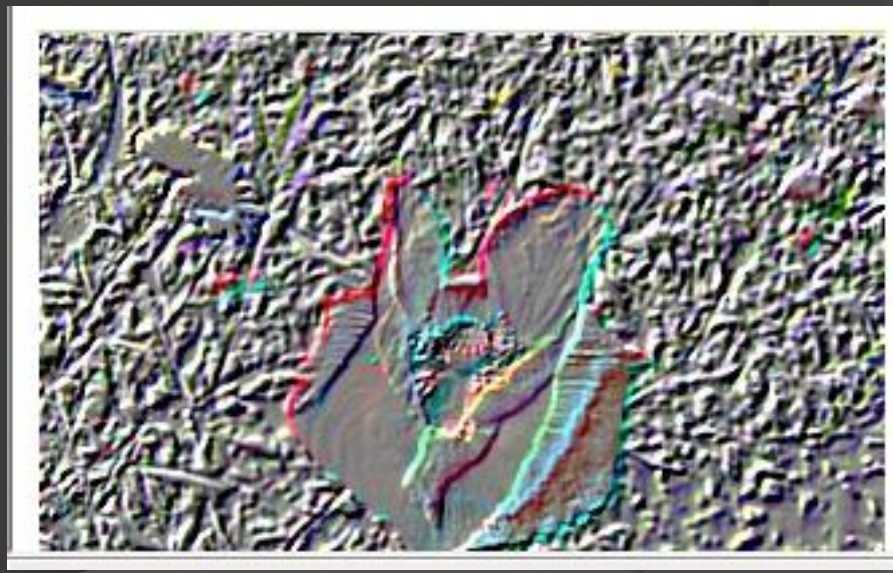
Görüntüyü keskinleştirmek kenar bulmaya çok benzer. Orijinal görüntü ve kenarları bulunmuş görüntü toplanır ve sonuçta kenarları kuvvetlendirilmiş yeni bir görüntü oluşur. Bu görüntüyü eklemek bir önceki örnekteki kenar bulma filtresinin merkez değeri bir arttırılarak yapılır. Filtre elemanlarının toplamı 1'dir ve sonuç orijinal görüntü ile aynı parlaklıkta fakat daha keskin olacaktır.

```
double filter[filterWidth][filterHeight] =
{
    -1, -1, -1,
    -1,  9, -1,
    -1, -1, -1
};

double factor = 1.0;
double bias = 0.0;
```

Kabartma yapma

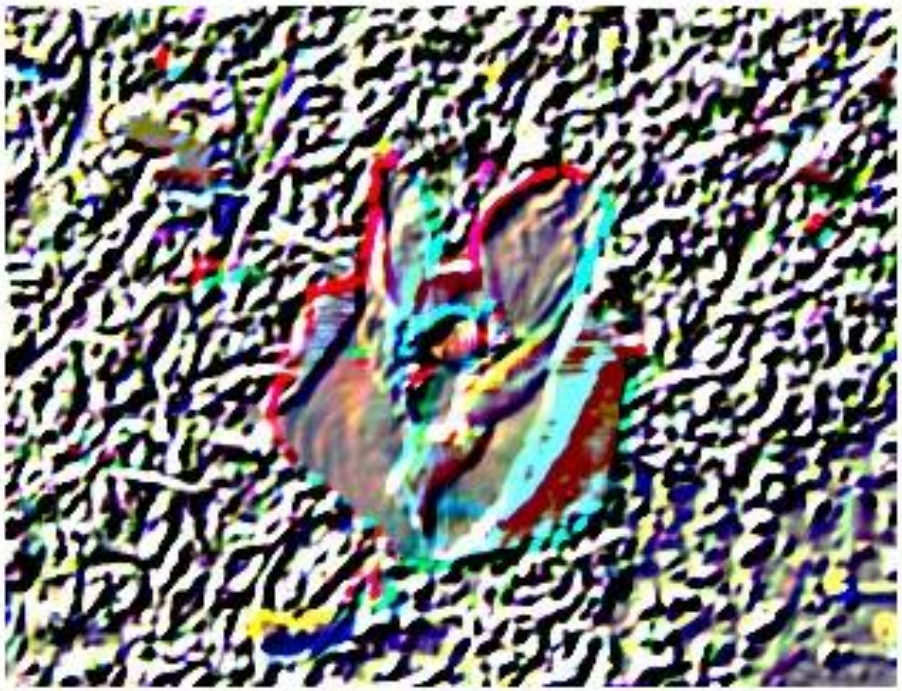
Bir kabartma filtresi bir görüntüye 3-boyutlu gölge efekti verir. Sonuç, görüntünün tümseklik haritası için çok faydalıdır. Bu, merkezin bir tarafındaki piksellerin alınarak diğer tarafındakilerden çıkarılması ile yapılabilir. Pikseller pozitif veya negatif sonuç değerleri alabilir. Negatif pikselleri gölge ve pozitif olanları aydınlık olarak kullanmak için 128 tabanı görüntüye eklenir. Artık görüntünün çoğu gri renge ve kenarlar koyu gri/siyah veya açık gri/beyaz'a dönüşecektir.
Aşağıda 45 derecelik kabartma filtresi görülmektedir.

```
double filter[filterWidth][filterHeight] =
{
    -1, -1,  0,
    -1,  0,  1,
     0,  1,  1
};

double factor = 1.0;
double bias = 128.0;
```



Aşağıda daha geniş bir kabartma filtresi görülmektedir:

```
double filter[filterWidth][filterHeight] =
{
    -1, -1, -1, -1,  0,
    -1, -1, -1,  0,  1,
    -1, -1,  0,  1,  1,
    -1,  0,  1,  1,  1,
     0,  1,  1,  1,  1
};

double factor = 1.0;
double bias = 128.0;
```



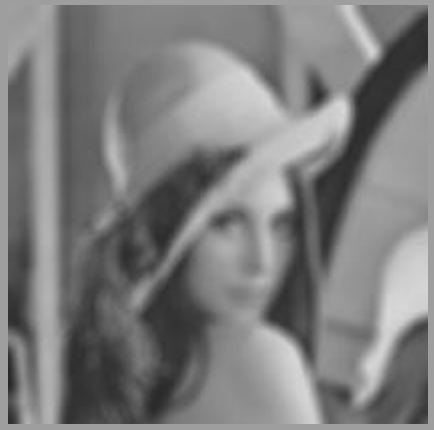
Matlab ile filtreleme

```
I = imread('C:\lena512.jpg');
subplot(2,2,1);
imshow(I); title('Orijinal Goruntu');

H = fspecial('motion',20,45);
HarekBulanik = imfilter(I,H,'replicate');
subplot(2,2,2);
imshow(HarekBulanik);title('Hareketli Bulanik Goruntu');

H = fspecial('disk',10);
Bulanik = imfilter(I,H,'replicate');
subplot(2,2,3);
imshow(Bulanik); title('Bulaniklastirilmis Goruntu');

H = fspecial('unsharp');
Keskin = imfilter(I,H,'replicate');
subplot(2,2,4);
imshow(Keskin); title('Keskinlestirilmis Goruntu');
```

Orijinal Goruntu

Hareketli Bulanik Goruntu

Bulaniklastirilmis Goruntu



Keskinlestirilmis Goruntu

Matlab ile kabartma yapma

```
h = fspecial('sobel');
I2 = filter2(h,I);
imshow(mat2gray(I2));
```