BSM409 Görüntü İşleme

Bölüm 5
Uzamsal Filtreleme

Dr. Öğr. Üyesi Caner ÖZCAN

If the facts don't fit the theory, change the facts.
~Einstein

İçerik

3. Yeğinlik Dönüşümleri ve Uzamsal Filtreleme
   - Temel Bazı Yeğinlik Dönüşüm Fonksiyonları
   - Histogram İşleme
   - Uzamsal Filtrelemenin Esasları
   - Uzamsal Yumuşatma Filtreleri
   - Uzamsal Keskinleştirme Filtreleri
   - Uzamsal Zenginleştirme Yöntemlerini Birleştirme
   - *Yeğinlik Dönüşümleri ve Uzamsal Filtreleme İçin Bulanık Tekniklerin Kullanılması*

Komşuluk Önemi

- Hem zebralarda hem de dalmaçyalılarda benzer sayılarda siyah ve beyaz pikseller vardır.
- İkisi arasındaki fark, tek piksel değerleri yerine küçük grup piksellerin karakteristik görünümüdür.

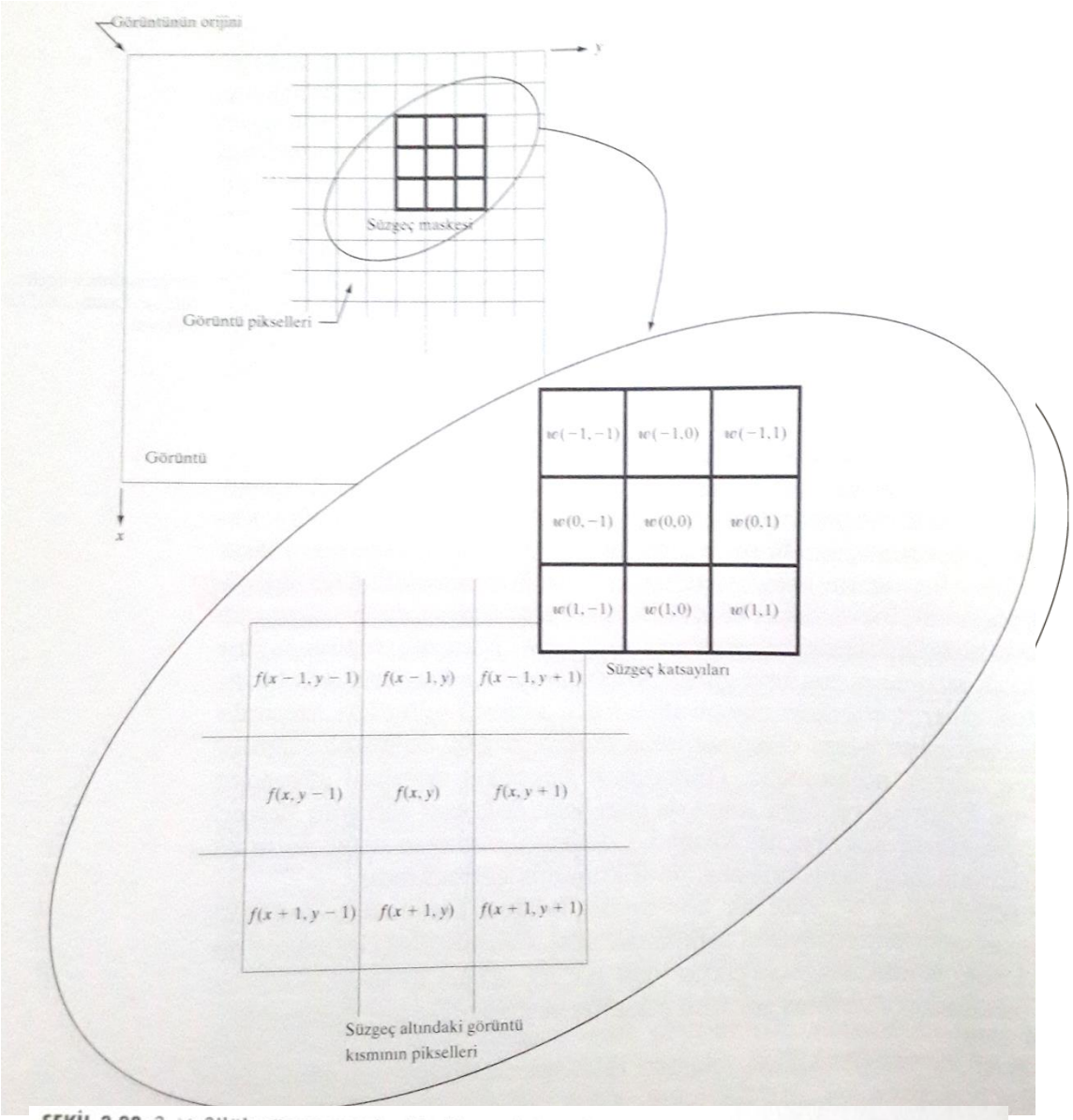
Uzamsal Filtreleme

- Uzamsal bir filtre (a) bir komşuluk bölgesinden ve (b) bir ön tanımlı işlemden oluşmaktadır.

- MxN'lik bir görüntünün mxn'lik bir filtre ile doğrusal uzamsal filtrelenmesi şu şekilde ifade edilir:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Uzamsal Filtreleme



ŞEKİL 3.28 3 × 3'lük süzgeç maskesi kullanan doğrusal uzamsal süzmenin işleyişi. Süzgeç maskesi katsayılarının koordinatlarını göstermek için seçilen biçim, doğrusal süzme için ifadelerin yazılmasını kolaylaştırmaktadır.

Uzamsal Filtreleme (2B Hareketli Ortalama)

$F[x, y]$

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								

Slide credit: S. Seitz

Uzamsal Filtreleme (2B Hareketli Ortalama)

$F[x, y]$

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							

Slide credit: S. Seitz

Uzamsal Filtreleme (2B Hareketli Ortalama)

$F[x, y]$

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						

Slide credit: S. Seitz

Uzamsal Filtreleme (2B Hareketli Ortalama)

$F[x, y]$

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					

Slide credit: S. Seitz

Uzamsal Filtreleme (2B Hareketli Ortalama)

$F[x, y]$

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				

Slide credit: S. Seitz

Uzamsal Filtreleme (2B Hareketli Ortalama)

$$F[x, y]$$

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	

Slide credit: S. Seitz

Uzamsal Yumuşatma Filtreleri

- Yumuşatma filtreleri, bulanıklaştırma ve gürültü azaltma amacıyla kullanılmaktadır.

- Bulanıklaştırma, görüntüdeki küçük detayların ortadan kaldırılması ve doğrular veya eğrilerdeki küçük boşlukların bağlanmasında kullanılır.

- Doğrusal ve doğrusal olmayan filtreleri içerir.

İki Yumuşatma (Ortalama Alma) Filtresi Maskesi

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

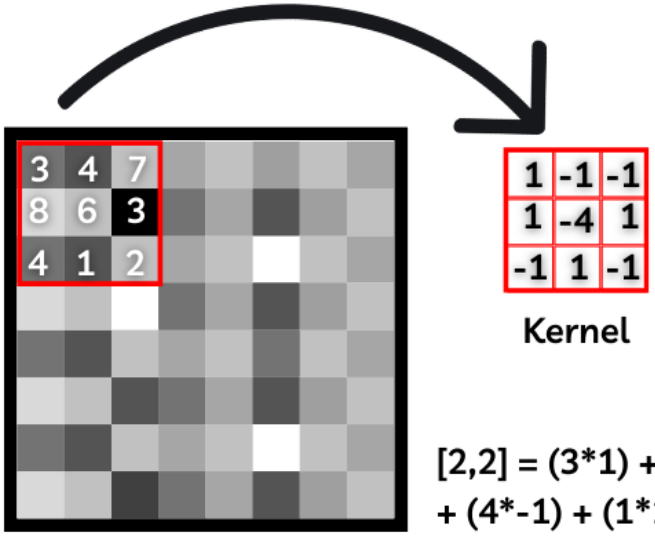
a b

ŞEKİL 3.32 3×3'lük iki yumuşatma (ortalama alma) süzgeç maskesi. Herbir maskenin önündeki çarpan, bir ortalama hesabında olması gerektiği gibi 1 bölü maske katsayılarının toplamı'na eşittir.

Python Kodu:

```
kernel1 = np.ones((3, 3))
img = cv2.filter2D(src=image, ddepth=-1, kernel=kernel1)
```

İki Yumuşatma (Ortalama Alma) Filtresi Maskesi



Image

1	-1	-1
1	-4	1
-1	1	-1

Kernel

3*1	4*-1	7*-1
8*1	6*-4	3*1
4*-1	1*1	2*-1

[2,2] = (3*1) + (4*-1) +(7*1) +(8*1) + (6*-4) + (3*1) + (4*-1) + (1*1) + (2*-1)

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

Identity kernel

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

Edge detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen kernel

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Box blur

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian blurr kernel

Bir görüntüyü 2 boyutlu evrişim matrisiyle bulanıklaştırma

```
# importing the modules needed
import cv2
import numpy as np

# Reading the image
image = cv2.imread('image.png')

# Creating the kernel(2d convolution matrix)
kernel1 = np.ones((5, 5), np.float32)/30

# Applying the filter2D() function
img = cv2.filter2D(src=image, ddepth=-1, kernel=kernel1)

# Shoeing the original and output image
cv2.imshow('Original', image)
cv2.imshow('Kernel Blur', img)

cv2.waitKey()
cv2.destroyAllWindows()
```

Yumuşatma (Ortalama Alma) Filtresi Maskesi

- Ortalama filtresini gerçekleştirmek için *cv2.blur()* ve *cv2.boxFilter()* işlevleri kullanılabilir.
- Her iki işlev de çekirdeği kullanarak bir görüntüyü düzgünleştirir.

Syntax of cv2.blur()

```
cv2.blur(image, ksize)
```

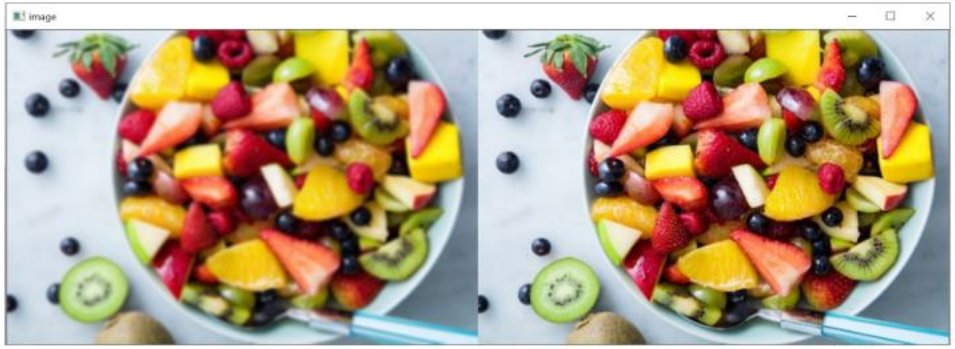
Syntax of cv2.boxFilter()

```
cv2.boxFilter(src, dst, depth, ksize, anchor, normalize, bordertype)
```

Yumuşatma (Ortalama Alma) Filtresi Maskesi

Example of Averaging Filter

```
import cv2
import numpy as np

# image path
path = r'salad.jpg'

# using imread()
img = cv2.imread(path)

im1 = cv2.blur(img,(5,5))
im2 = cv2.boxFilter(img, -1, (2, 2), normalize=True)

cv2.imshow('image', np.hstack((im1, im2)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```



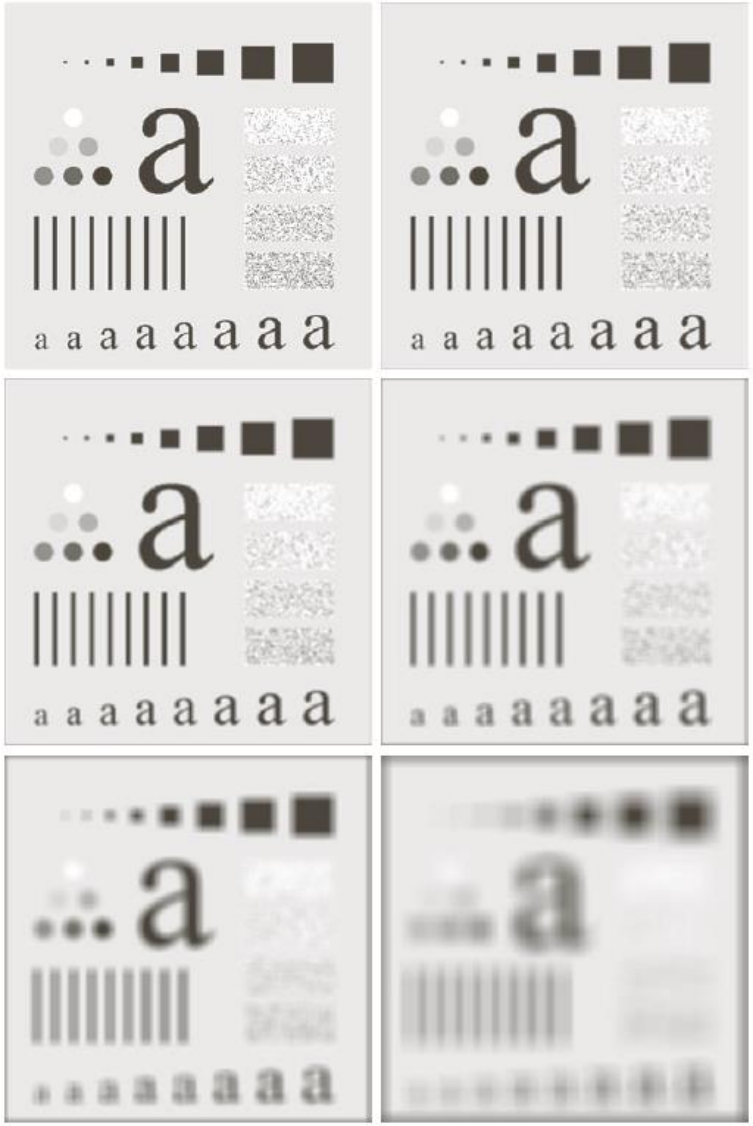
Doğrusal Yumuşatma Filtreleri

MxN'lik bir görüntünün mxn'lik bir ağırlıklı ortalama alma filtresi ile filtrelenmesi işlemi şu ifade ile verilir:

$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

burada $m = 2a+1, \quad n = 2b+1.$

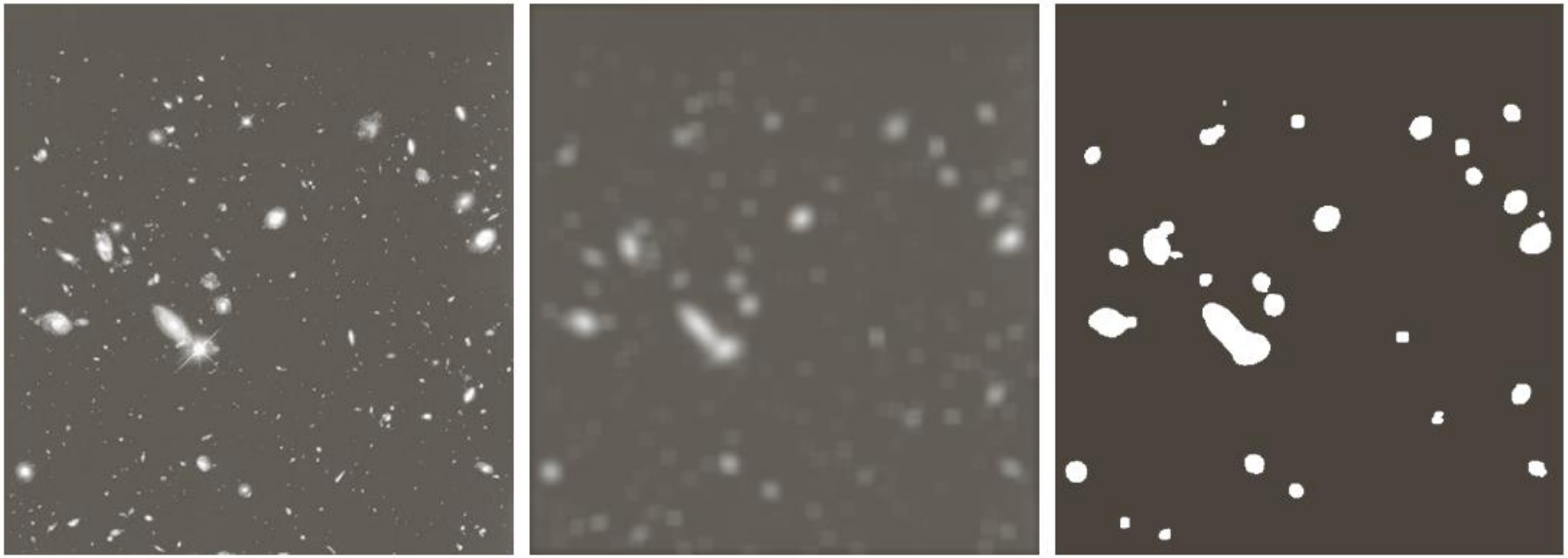
Doğrusal Yumuşatma Filtreleri



ŞEKİL 3.33 (a) Boyutları 500 × 500 piksel olan orijinal görüntü (b)-f) Sırasıyla boyutları $m = 3, 5, 9, 15$ ve 35 olan kare ortalama sügeçler ile yumuşatmanın sonuçları. Alttaki harflerin boyutları 2 punto artışlarla 10 ile 24 punto arasındadır. Üstteki büyük harf ise 60 puntodur. Dikey çubuklar 5 piksel genişliğinde ve 100 piksel yüksekliğindedir;aralıkları 20 pikseldir. Çemberlerin çapı 25 pikseldir ve sınırlarının arası 25 pikseldir;yeğinlik seviyeleri, %20 siyahlık artışıyla %0'dan %100'ye gitmektedir. Görüntünün arkaplanı %10 oranında siyahtır. Gürültülü dikdörtgenlerin boyutu 50 × 120 pikseldir.

a b
c d
e f

Örnek: Nesnelerin Kaba Temsili



a b c

ŞEKİL 3.34 (a) Hubble uzay teleskopundan gelen 528 × 485 piksel boyutuna sahip görüntü. (b) 15 × 15'lik bir ortalama alma maskesi ile süzülmüş görüntü. (c) (b)'nin eşikleme yapılmış sonucu. (Orijinal görüntü NASA'nın izniyle)

Sıra İstatistiği (Doğrusal Olmayan) Filtreler

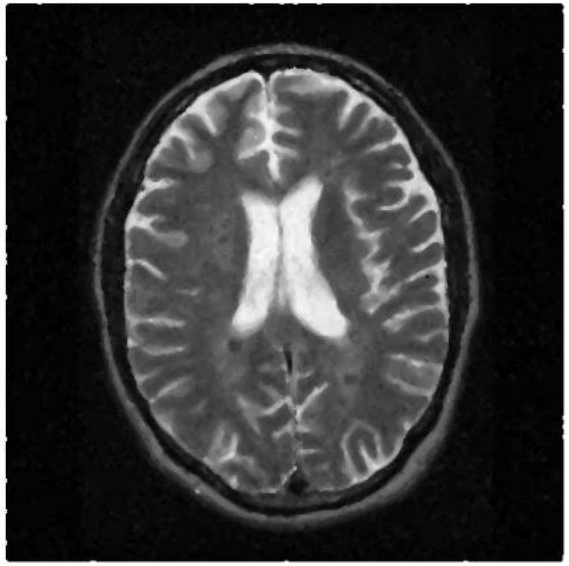
— Doğrusal olmayan filtrelerdir.

— Süzgeç tarafından çevrelenen görüntü bölgesindeki piksellerin sıralanması ve sonra da merkezdeki piksel değerinin bu sıralama sonucuyla tespit edilen değerle değiştirilmesidir.

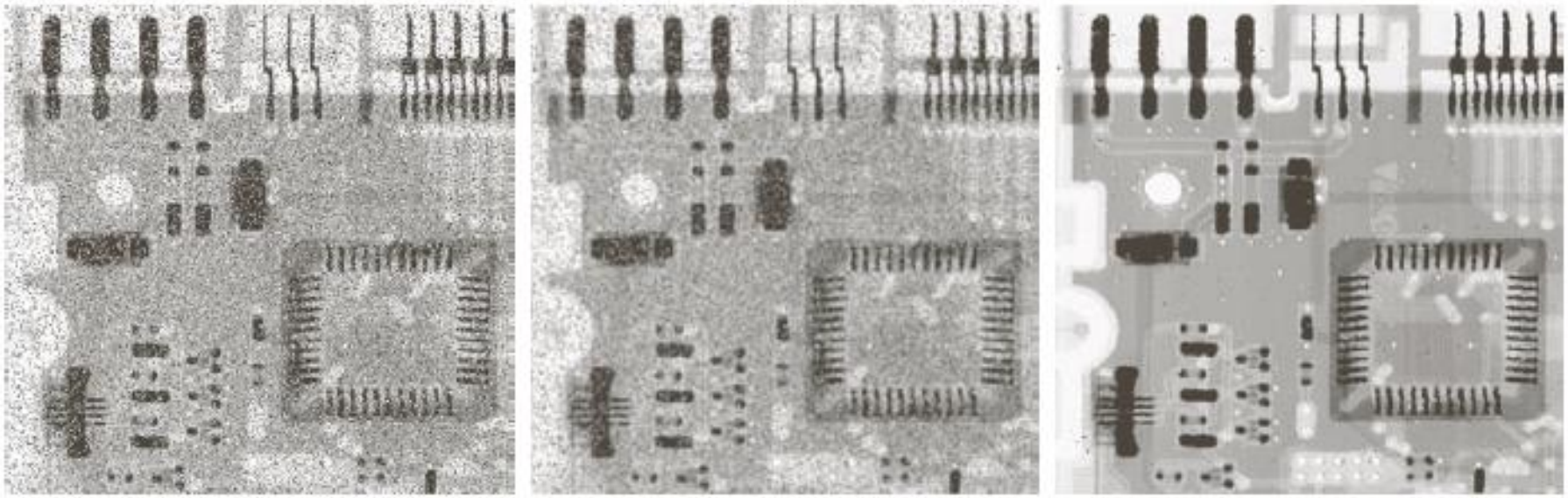
Örneğin medyan filtre, max filtre, min filtre

Sıra İstatistiği (Doğrusal Olmayan) Filtreler

```
import cv2
import numpy as np

img = cv2.imread('brain.jpg')
median = cv2.medianBlur(img, 5)
compare = np.concatenate((img, median), axis=1) #side by side comparison

cv2.imshow('img', compare)
cv2.waitKey(0)
cv2.destroyAllWindows
```



Örnek: Gürültü Giderimi İçin Medyan Filtre Kullanımı



a b c

ŞEKİL 3.35 Şekil 3.35 (a) Tuz-biber gürültüsüne maruz kalmış devre kartının X-ışını görüntüsü (b) 3×3'lük ortalama maske ile gürültünün azaltılmış hali (c) 3×3'lük ortanca süzgeç ile gürültünün azaltılmış hali

Python Kodu (medyan):

medianBlur(source_image, kernel_size)

Uzamsal Keskinleştirme Filtreleri

- Amaç: yeğinlikteki geçişleri vurgulamaktır.
- Laplas İşleci
- Keskin Olmayan Maskeleme ve Yüksek Vurgulu Filtreleme
- Doğrusal Olmayan Görüntü Keskinleştirmede Birinci Derece Türevlerin Kullanılması — Gradyan

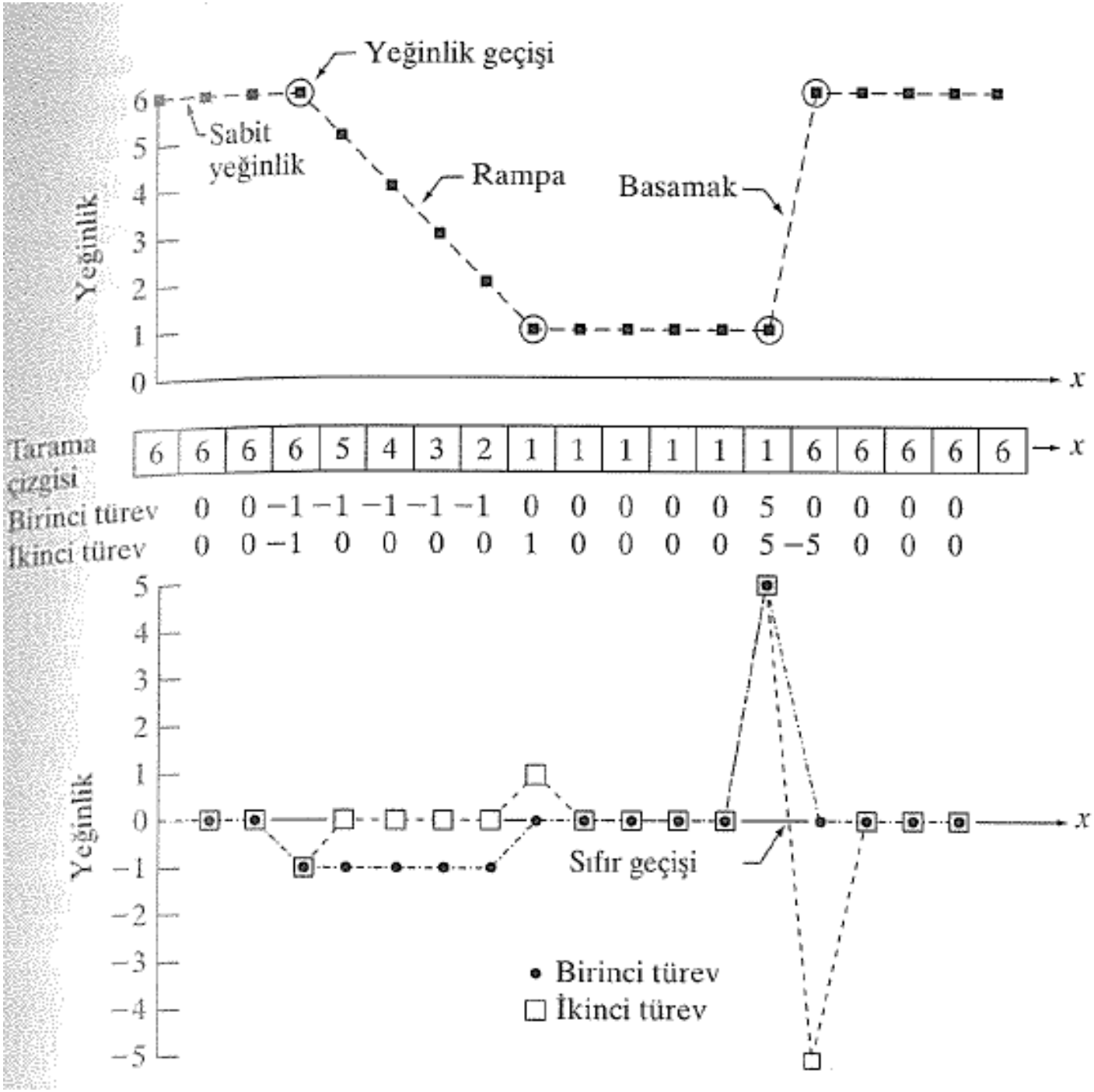
Uzamsal Keskinleştirme Filtreleri: Altyapı

- Bir boyutlu bir f(x) fonksiyonunun birinci derece türevinin temel tanımı, aşağıdaki farktır:

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

- f(x)'in ikinci dereceden türevini, fark olarak şöyle tanımlarız:

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



a
b
c

ŞEKİL 3.36
1-D sayısal bir fonksiyonun birinci ve ikinci dereceden türevlerinin gösterimi. Görüntüden elde edilen yatay bir yeğinlik kesitinin bir kısmı gösterilmektedir. (a) ve (c)'deki noktalar görüntüleme amaçlı olarak kesikli çizgiler ile birleştirilmiştir.

Uzamsal Keskinleştirme Filtreleri: Laplas İşleci

- İki değişkenli bir fonksiyon (görüntü) f(x,y) için Laplas işleci şu şekilde tanımlanır:

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

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

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

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

Uzamsal Keskinleştirme Filtreleri: Laplas İşleci

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

ŞEKİL 3.37
(a) Eşitlik (3.6-6)'yı gerçekleştirmek için kullanılan süzgeç maskesi.
(b) Bu eşitliğin, köşegen terimleri içerecek şekilde genişletilmiş halinin gerçekleştirilmesi için kullanılan maske.
(c) ve (d) Laplas işlecinin pratikte sıkça kullanılan diğer iki uyarlaması.

Uzamsal Keskinleştirme Filtreleri: Laplas İşleci

- Görüntü keskinleştirmek için Laplas işlecini şu şekilde kullanırız:

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

burada

$f(x,y)$ giriş görüntüsü

$g(x,y)$ keskinleştirilmiş görüntü

c parametresi seçilen filtrenin durumuna göre -1 veya 1 olur.

Uzamsal Keskinleştirme Filtreleri: Laplas İşleci

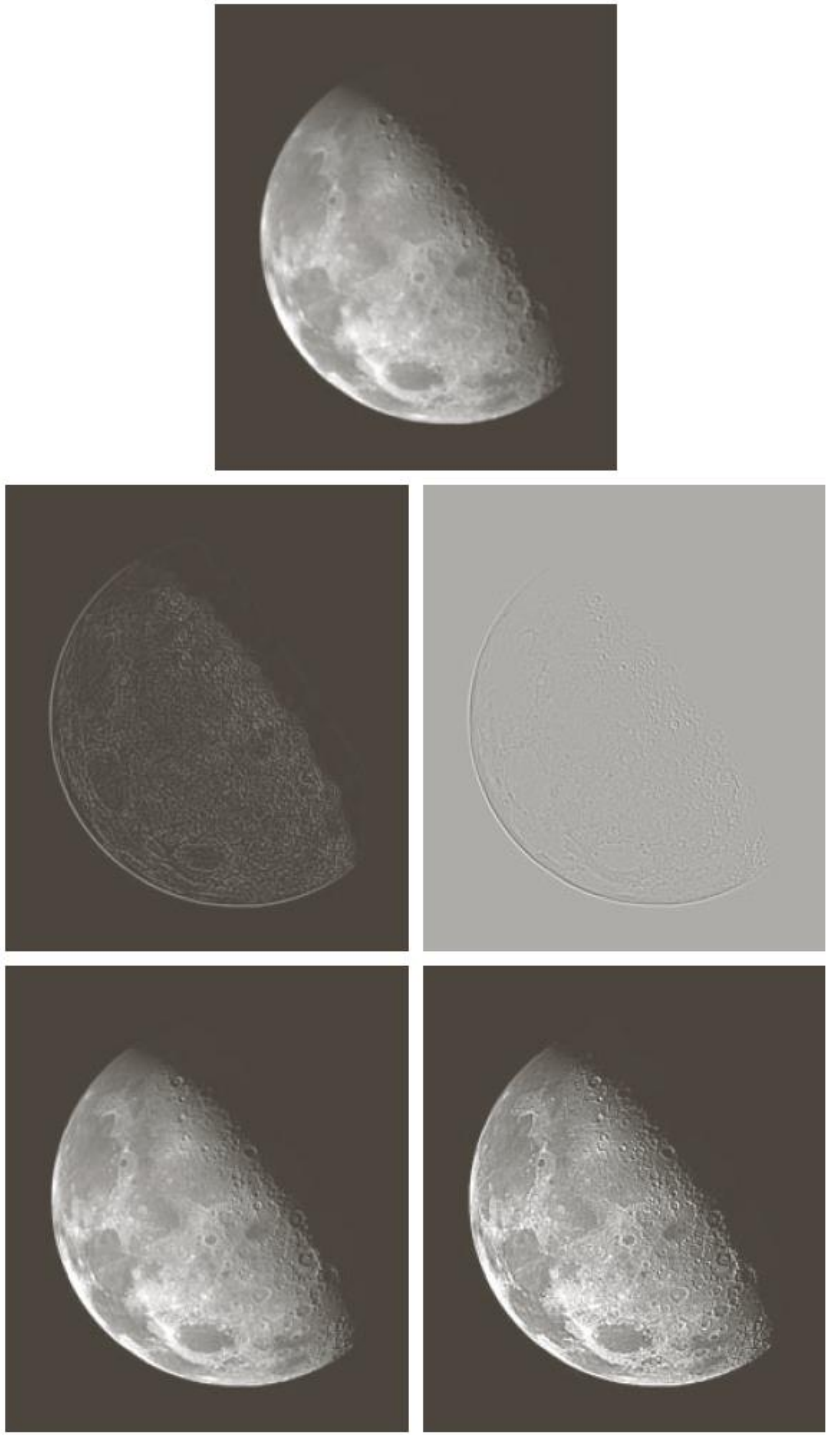
Python Kodu:

Laplacian(src_gray, dst, ddepth, kernel_size, scale, delta, BORDER_DEFAULT);

- src_gray: Giriş görüntüsü.
- dst: Hedef (çıkış) görüntüsü
- ddepth: Hedef görüntünün derinliği. Girdimiz CV_8U olduğundan, taşmayı önlemek için ddepth = CV_16S olarak tanımlarız.
- kernel_size: Dahili olarak uygulanacak Sobel operatörünün çekirdek boyutu. Bu örnekte 3 kullanıyoruz.
- scale, delta ve BORDER_DEFAULT: Varsayılan değerler olarak bırakıyoruz.

Uzamsal Keskinleştirme Filtreleri: Laplas İşleci

```python
"""
@file laplace_demo.py
@brief Sample code showing how to detect edges using the Laplace operator
"""
import sys
import cv2 as cv

def main(argv):
    # [variables]
    # Declare the variables we are going to use
    ddepth = cv.CV_16S
    kernel_size = 3
    window_name = "Laplace Demo"
    # [variables]

    # [load]
    imageName = argv[0] if len(argv) > 0 else 'lena.jpg'

    src = cv.imread(cv.samples.findFile(imageName), cv.IMREAD_COLOR) # Load an image

    # Check if image is loaded fine
    if src is None:
        print ('Error opening image')
        print ('Program Arguments: [image_name -- default lena.jpg]')
        return -1
    # [load]

    # [reduce_noise]
    # Remove noise by blurring with a Gaussian filter
    src = cv.GaussianBlur(src, (3, 3), 0)
    # [reduce_noise]

    # [convert_to_gray]
    # Convert the image to grayscale
    src_gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
    # [convert_to_gray]

    # Create Window
    cv.namedWindow(window_name, cv.WINDOW_AUTOSIZE)

    # [laplacian]
    # Apply Laplace function
    dst = cv.Laplacian(src_gray, ddepth, ksize=kernel_size)
    # [laplacian]

    # [convert]
    # converting back to uint8
    abs_dst = cv.convertScaleAbs(dst)
    # [convert]

    # [display]
    cv.imshow(window_name, abs_dst)
    cv.waitKey(0)
    # [display]

    return 0

if __name__ == "__main__":
    main(sys.argv[1:])
```



a
b c
d e

ŞEKİL 3.38
(a) Ay'ın Kuzey Kutbunun bulanık görüntüsü (b) Ölçeklemesiz Laplas görüntüsü (c) Ölçeklemeli Laplas görüntü. (d) 3.37(a)'daki maske kullanılarak keskinleştirilmiş görüntü. (e) Şekil 3.37(b)'deki maske kullanılarak elde edilen sonuç (Orijinal görüntü NASA'nın izniyle).

Birinci Derece Türevlere Dayalı Görüntü Keskinleştirme

$f(x, y)$ fonksiyonu için (x, y) koordinatlarındaki f'in gradyanı iki boyutlu sütun vektörü olarak tanımlanır:

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

∇f vektörünün, $M(x, y)$ olarak gösterilen büyüklüğü:

Gradyan görüntü $$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^{\,2} + g_y^{\,2}}$$

Birinci Derece Türevlere Dayalı Görüntü Keskinleştirme

∇f vektörünün, $M(x, y)$ olarak gösterilen büyüklüğü:

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^{\,2} + g_y^{\,2}}$$

$$M(x, y) \approx |g_x| + |g_y|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$M(x, y) = |z_8 - z_5| + |z_6 - z_5|$$

Birinci Derece Türevlere Dayalı Görüntü Keskinleştirme

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

a
b d
c e

ŞEKİL 3.41
(a) Bir görüntünün 3×3'lük bölgesi (z'ler yeğinlik değeridir). (b)-(c) Roberts çapraz gradyan operatörleri. (d)-(e) Sobel operatörleri. Türev operatörlerinden beklenen şekilde tüm maske katsayıları toplamı sıfırdır.

Birinci Derece Türevlere Dayalı Görüntü Keskinleştirme

Roberts çapraz gradyan operatörleri

$$M(x,y) \approx |z_9 - z_5| + |z_8 - z_6|$$

Sobel operatörleri

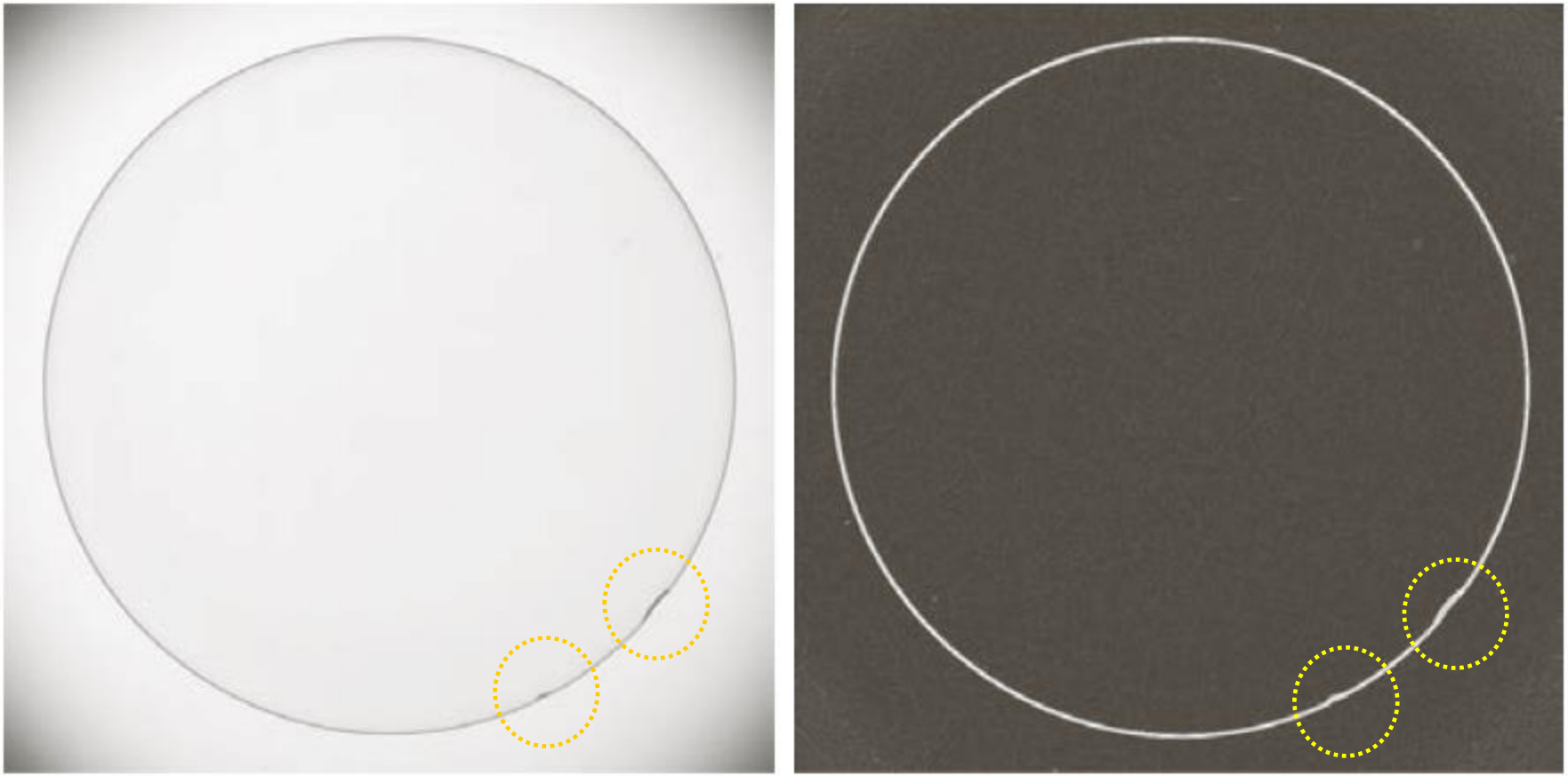
$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Örnek

a b

ŞEKİL 3.42
(a) Kontak lensin optik görüntüsü (saat 4 ve 5 yönündeki sınırlarda bulunan kusurlara dikkat ediniz). (b) Sobel gradyanı (Orijinal görüntü Pete Sites, Perceptics Corporations izniyle)

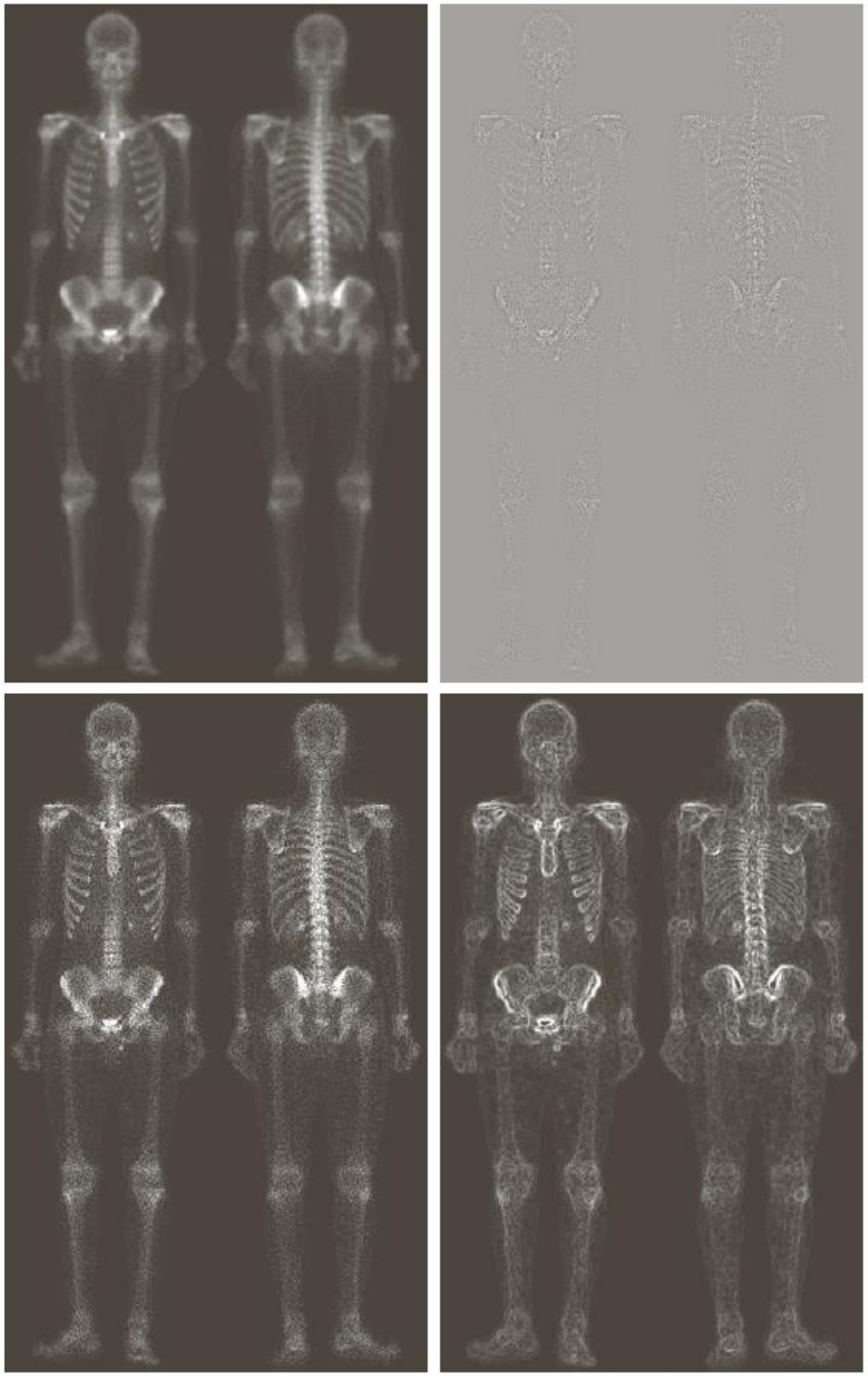


Örnek

Uzamsal Zenginleştirme Yöntemlerini Birleştirme

Amaç:

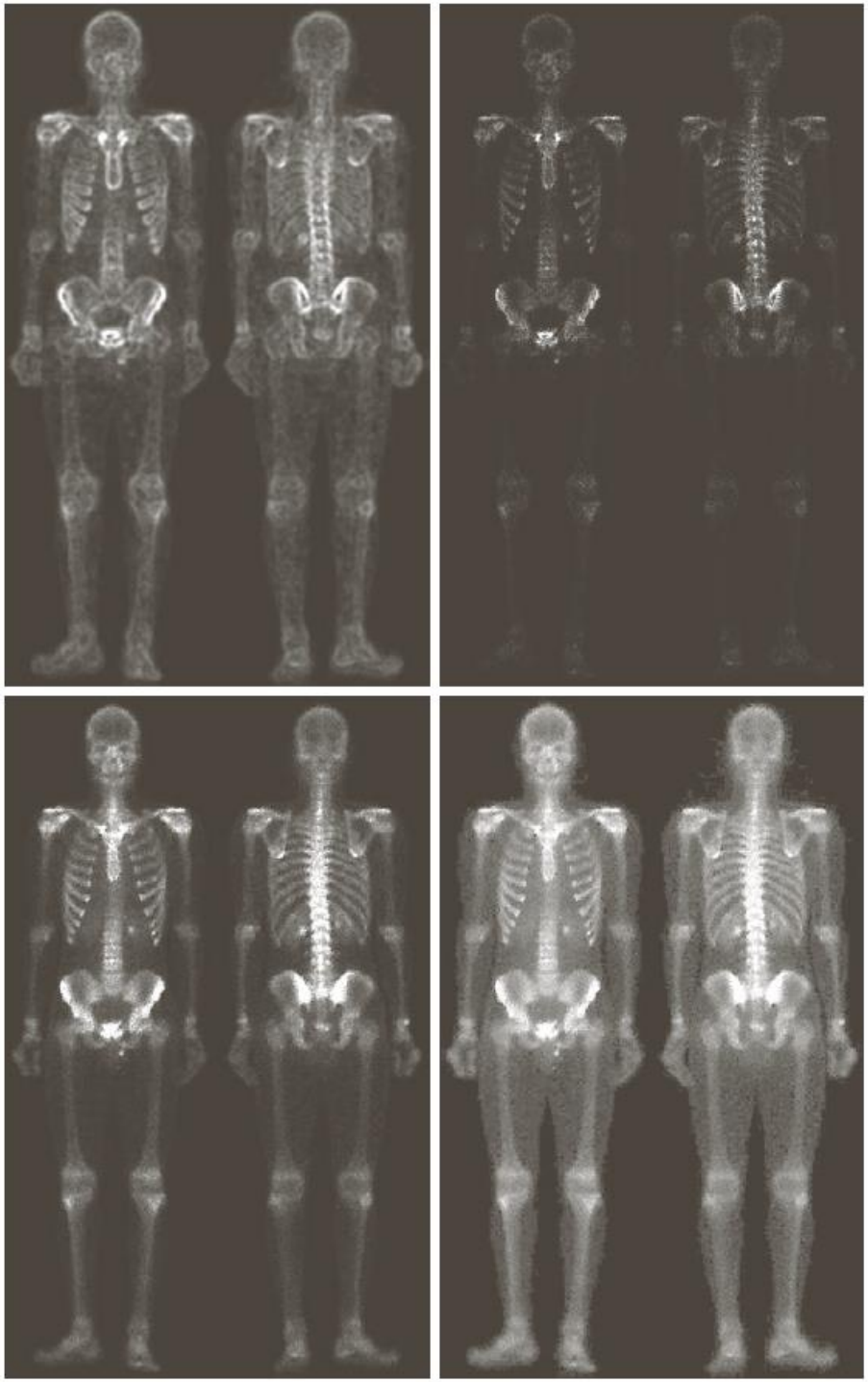
Görüntüyü keskinleştirerek ve iskelet ayrıntılarının çoğunu ortaya çıkararak zenginleştirmektir.



a b
c d

ŞEKİL 3.43
(a) Tam vücut kemik taramasının görüntüsü, (b) (a)'nın Laplası. (c) (a) ve (b)'nin toplanmasıyla elde edilen keskinleştirilmiş görüntü. (d)(a)'nın Sobel gradyanı.

Örnek



e f
g h

ŞEKİL 3.43
(devam ediyor)
(e) (e) 5 × 5'lik bir ortalama alma süzgeci ile yumuşatılmış Sobel görüntü. (f) (c) ve (e)'nin çarpımı ile oluşturulan maske görüntüsü. (g) (a) ve (b)'nin toplamı ile elde edilen keskinleştirilmiş görüntü. (h) Bir kuvvet kanunu dönüşümünün (g)'ye uygulanmasıyla elde edilen nihai sonuç. (g) ve (h)'yi (a) ile karşılaştırınız (Orijinal görüntü G.E. Medical Systems izniyle).

Kaynaklar

- Sayısal Görüntü İşleme, Palme Yayıncılık, Üçüncü Baskıdan Çeviri (*Orj: R.C. Gonzalez and R.E. Woods: "Digital Image Processing", Prentice Hall, 3rd edition, 2008*).
- “Digital Image Processing Using Matlab”, Gonzalez & Richard E. Woods, Steven L. Eddins, Gatesmark Publishing, 2009
- Ders Notları, CS589-04 Digital Image Processing, F.(Qingzhong) Liu, http://www.cs.nmt.edu/~ip
- Ders Notları, BIL717-Image Processing, E.Erdem
- Ders Notları, EBM537-Görüntü İşleme, F.Karabiber
- https://docs.opencv.org/
- https://www.geeksforgeeks.org/python-opencv-filter2d-function/
- https://medium.com/@florestony5454/median-filtering-with-python-and-opencv-2bce390be0d1
- Bekir Aksoy, Python ile İmgeden Veriye Görüntü İşleme ve Uygulamaları, Nobel Akademik Yayıncılık