22083

İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL GÖRÜNTÜLERDE KENAR TANIMA METODLARI

YÜKSEK LİSANS TEZİ
Elk. Müh. ALTUĞ ERDÖN

Tezin Enstitüye Verildiği Tarih : 22 Haziran 1992
Tezin Savunulduğu Tarih : 22 Eylül 1992

Tez Danışmanı : Doç. Dr. Ahmet H. Kayran
Diğer Jüri Üyeleri : Prof. Dr. Cem Göknar
Doç. Dr. Melih Pazarcı

EYLÜL 1992

ÖNSÖZ

Bu tezde görüntü işleme konusunda ana konulardan birisi olan kenar tanıma yöntemleri incelenmiş ve gürültülü ortamlardaki performansları karşılaştırılmıştır.

Bu tezin hazırlanmasında değerli yardımlarını esirgemeyen sayın hocam Doç. Dr. Ahmet H. Kayran'a teşekkür ederim.

EYLÜL 1992 ALTUĞ ERDÖN

İÇİNDEKİLER

ŞEKİL LİSTESİ		V
ÖZET		VII
SUMMARY		VIII
BÖLUM 1.	GİRİŞ	1
BÖLUM 2.	GÜRÜLTÜ AZALTMA TEKNİKLERİ	4
2.1	Medyan Filtre	4
BÖLUM 3.	TUREV METODLARI	7
3.1.1.	Gradyent Metodları	8
3.1.2.	Kompass Kenar Oparatörü	13
3.2.1.	Laplasyan Kenar Oparatörü	17
3.2.2.	Laplasyan-Gauss Metodu	19
3.3	Sonuç	25
BÖLÜM 4.	YÖRESEL İSTATİSTİK METODU	27
4.1	Pencereleme Basamağı	27
4.2	Kenar Piksellerinin Bulunması	29
4.3	Sonuç	30
BÖLÜM 5.	STOKASTİK GRADYENT METODU	32
5.1.	2-B Fonksiyonların Stokastik Modelleri	32
5.2.1	Wiener Filtresi	33
5.2.2	Stokastik Gradyent Maskesi	35
5.2.3	Korolasyon Matrisinin Bulunması	37
5.3	Sonuç	37
BÖLÜM 6.	FIR KENAR TANIMA FİLTRESİ	40
6.1	Filtrenin Tanımlanması	40
6.2	Filtre Transfer Fonksiyonunu Bulunması	41
6.3	Filtrenin Gerçeklemesi	43
6.4	Sonuç	45

BÖLÜM 7 MORFOLOJİK KENAR TANIMA METHODU 49
7.1 Temel Morfolojik Operasyonlar 49
7.2 Morfolojik Kenar Tanıma Operatörü 50
7.3 Minimum ve Bulanık Kenar Tanıma Operatörü 54
7.4 Sonuç 55
BÖLUM 8. KENAR TANIMA METODLARININ KARŞILAŞTIRILMASI 58
SONUÇLAR VE ÖNERİLER 69
KAYNAKLAR 70
EK1 BİLGİSAYAR PROGRAMLARININ AÇIKLANMASI 71
EK2 BİLGİSAYAR PROGRAMLARI 73
ÖZGEÇMİŞ 102

Şekil Listesi

Şekil 1.1 Değişen gri seviyeli E harfi
Şekil 1.2 Çeşitli kenar fonksiyonları
Şekil 2.1 (+) tipinde 5x5 medyan filtre
Şekil 2.2 Sobel yöntemi kenar görüntüsü (eşik değeri=0)
Şekil 2.3 5x5 (+) tipi medyan filtrelemeden sonraki Sobel yöntemikenar görüntüsü (eşik değeri=0)
Şekil 3.1 Rampa kenarın 1. ve 2. türevleri
Şekil 3.2 Gradyent metodu blok diyagramı
Şekil 3.3 Yönlü kenar maskeleri
Şekil 3.4 Yönsüz kenar maskeleri kenar görüntüleri
Şekil 3.5 Yönsüzkenar maskeleri
Şekil 3.6 Yönlü kenar maskeleri kenar görüntüleri
Şekil 3.7 Kompas maskeleri
Şekil 3.8 Kompas maskesi kenar görüntüsü
Şekil 3.9 Laplasyan kenar maskeleri
Şekil 3.10 Laplasyan metodu kenar görüntüleri
Şekil 3.11 Varyans testli Laplasyan metodu kenar görüntüleri
Şekil 3.12 Laplasyan metodu blok diyagramı
Şekil 3.13 LoG filtresikenar görüntüleri
Şekil 3.14 Izgara test resmi
Şekil 3.15 Izgara test resmi kenar görüntüsü
Şekil 4.1 Bazı kenar tiplerinin yöresel ortalamalarının değişimi
Şekil 4.2 Görüntünün pencerelere bölünmesi
Şekil 4.3 Yöresel istatistik metodu blok diyagramı
Şekil 4.4 Yöresel istatistik metodu kenar görüntüleri
Şekil 5.1 1 piksel geçişli kenar modeli
Şekil 5.2 Dikey stokastik gradyent maskesi
Şekil 5.3 Stokastik gradyent metodu kenar görüntüleri
Şekil 6.1 FFT metodu ile gerçekleştirme blok diyagramı
Şekil 6.2 FIR filtre metodu kenar görüntüsü , c=1.5
Şekil 6.3 FIR filtre metodu kenar görüntüsü , c=3.0
Şekil 6.4 FIR filtre metodu kenar görüntüsü , c=5.0
Şekil 7.1 Bir piksel gürültülü görüntü ve kenar görüntüsü
Şekil 7.2 Değişik yönlerde kenar görüntüleri
Şekil 7.3 Maksimum operatörü kenar görüntüsü
Şekil 7.4 Toplam operatörü kenar görüntüsü
Şekil 7.5 Minimum operatörü kenar görüntüsü
Şekil 7.6 Minimum ve bulanık operatörü kenar görüntüsü
Şekil 8.1 Sobel metodu kenar görüntüsü, SNR=1
Şekil 8.2 Kompas metodu kenar görüntüsü, SNR=1
Şekil 8.3 Laplasyan metodu kenar görüntüsü, SNR=1
Şekil 8.4 LoG metodu kenar görüntüsü , SNR=1
Şekil 8.5.a Yöresel İstatistik Metodu , SNR=1
Şekil 8.5.b Yöresel İstatistik Metodu, SNR=1
Şekil 8.6.a Stokastik gradyent metodu , SNR=1

Şekil 8.6.b Stokastik gradyent metodu, SNR=1
Şekil 8.7 FIR filtre kenar görüntüsü, SNR=1 , c=3.5
Şekil 8.8.a Morfolojik metod, SNR=1
Şekil 8.8.b Morfolojik metod, SNR=1
Şekil 8.9 Sobel metodu, gürültü oranı-hata yüzdesi grafiği
Şekil 8.10 Log filtresi gürültü oranı-hata yüzdesi grafigi
Şekil 8.11 Yöresel istatistik metodu gürültü oranı-hata yüzdesi grafiği
Şekil 8.12 Stokastik gradyent metodu gürültü oranı-hata yüzdesi grafiği
Şekil 8.13 FIR filtresi , gürültü oranı-hata yüzdesi grafiği
Şekil 8.14 Morfolojik metodu, gürültü oranı-hata yüzdesi grafiği

ÖZET

Görüntü işleme konusunda temel öneme sahip konulardan birisiide kenar tanımadır. Görüntülerin gri seviyelerindeki ani değişikliklerin olduğu bölgelere kenar denir. Cisimlerin fiziksel özellikleri ile kenarları arasında direkt bir ilişki vardır, görüntünün birçok fiziksel özelliği kenarlarından çıkarılabilir. Bu nedenle görüntü işleme sistemlerinin ilk basamaklarında kenar tanıma işlemleri önem taşımaktadır.

Kenar tanıma metodları gri seviyeli görüntüleri kenar görüntülerine çevirmektedir. Bu değişim orijinal görüntüdeki birçok yararlı fiziksel özelliği değiştirmeden aktarır. Bundan sonraki görüntü işleme basamaklarına basit formlara indirgenmiş verilerin uygulanmasına imkan verir.

Bu tezde genel kenar tanıma metodları incelenmiştir. En yaygın kullanım olan türev yöntemleri içerik olarak oldukça eksik olmalarına rağmen hale kullanımdadır ve hale teorilerinin geliştirilmesine çalışılmaktadır. Yöresel ortalama değişimleri incelenerek istatistiksel testle kenar bulunması incelenen diğer bir konudur. Gürültüyüde hesaba katan stokastik modeller kullanarak yapılan Wiener filtresi ile kenar tanıma metodu özellikle gürültülü ortamalar için incelediğimiz diğer bir bölümdür. FIR filtre kullanılarak yapılan enerjiyi maksimize etmeye çalışan yöntem filtreleme yöntemlerini incelediğimiz bölümdür. Son yöntem olarakta yeni bir konu olan morfolojik kenar operatörü incelenmiştir. Son bölümde de buraya kadar incelenen metodların genel bir karşılaştırılması yapılacak ve performasları incelenecektir.

SUMMARY
EDGE DETECTION

A problem of fundamental importance in image analysis is edge detection. Edge points can be thought of as pixel locations of abrupt gray-level change. As there is a direct relationship between the edges and the physical properties of a scene , much of the scene information can be recovered from an edge image. Thus edge detection is a key step in the early processing of a computer vision system. Edge detection converts gray scale image into a binary edge image which may have direction information. The transformation preserves a great deal of the useful information in the orijinal image. The rest of the vision processes can deal with the simple form, instead of dealing with gray scale image information. Therefore edge detection has been a topic of fruitful research in recent years.

5 different approach to the problem are analyzed. They can be grouped in the following categories.

1. Differentiation techniques
2. Local statistical technique
3. Filtering technique
4. Stochastic gradient technique
5. Morphological technique

Abrupt changes in gray level can take several different forms. The most common is the step edge. This is , of course, the ideal case. A step edge seperates two regions in each of which the gray level is relatively uniform with different values on the two sides of the edge. But the presence of blur and noise turns steps into noisy ramps edges which have a transition region between two different gray levels. Another important type of gray level discontinuity is the line, which is a thin strip that differs from the regions on both sides of it. Also abrupt changes in the rate of change of gray can also be defined as edges.

A difficulty with edge detection is that the detected edges often have gaps in them at places where the transitions between regions are not abrupt enough. Moreover, edges may be detected at points that are not part of region boundaries, if the given picture is noisy.

For a continues image $f(x,y)$ its derivative assumes, a local maximum in the direction of the edge.

Threfore one of the most fruitful ways to produce an edge image based on differentiation . At this edge detection techniques , the gradient of the input image is measured at a specific direction and, then magnitude of the gradient thresholded with a appropriate threshold value. Performance of this method depends on both choice of gradient type and threshold value.

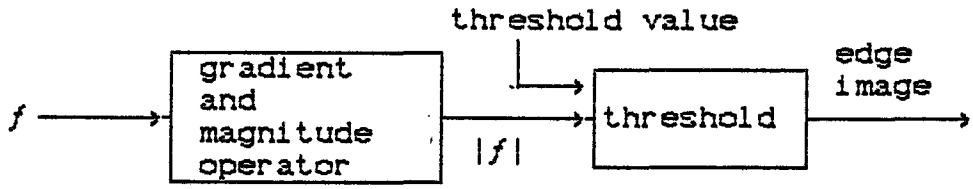


figure: Procedure for producing edge images

Based on this gradient method two types of edge detection developed. These are gradient operators and compass operators.

Gradient operators can be classfied into two system. Directional and non-directionl operators. for example, if the system uses $|\partial f(x,y)/\partial x|$ this system can detect edges at vertical direction, but can not detect edges at other directions. Since this function that has a bias toward one particulr direction, it is called directional edge detector. If the system uses $|\nabla f(i,j)|$ since this system do not have a bias toward any particular direction, this method called non-directional detector. For digital images the operators also called mask . These mask represents finite difference approximation of directional masks.

Once the magnitude of the gradient calculated , pixel location (m,n) is declared as an edge point if gradient at (m,n) exceeds some threshold t. The locations of edge points constitude an edge map , e(m,n) which is defined as

$$e(m,n) = \begin{cases} 1 & , \quad (m,n) \in I_g \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$I_g = \{(m,n); g(m,n) > t\}$$

Edge map gives the necessary data for tracing the object boundries in an image. Typically , t may be selected using the cumulative histogram of g(m,n) so that pixel magnitude of which 5 to 10% of the largest gradient magnitude are declared as edges. Some of the common gradient operators are Prewitt, Robert, Isotropic and Sobel.

Compass operators measure gradients in a selected number of directions . An counterclockwise circular shift of the eight boundary elements of these mask gives a 45° rotation of the gradient direction.

Another frequently encountered operator is the Laplacian , defined as

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

Because of the second order derivatives , this gradient operator is more sensitive to noise than those previously defined. Also, thresholded magnitude produce double edges. For thee reasons, together with inability detect edge directions, the Laplacian as such is not a good edge detection algoritms.

A better utilization of the Laplacian operator, Laplacian of Gaussian functions is a powerfull zero crossing. Laplacian of Gaussian functions is defined as

$$\nabla^2 h(x,y) = \frac{e^{-(x^2+y^2)/(2\pi\sigma^2)}}{(\pi\sigma^2)^2} \left(x^2 + y^2 - 2\pi\sigma^2 \right)$$

where σ controls the width of the Gaussion kernel and k normalizes the sum of the elements of a given size mask to unity. Such kind of zero crossing edge detector is equivalent to a low pass filter having a Gaussian impulse response followed by a Laplace operator. The low pass filter serves to noise sensitivity of the Laplacin .Laplacian of Gaussian is notated as LoG.. LoG filters was, first, defined by Marr and Hilbert.

Previous derivative methods fails in the noisy pictures. Using local average changes instead of gray values gives better performance in the noisy pictures. But, using local averages detect edges very abruptly. At this method, local averages of small windows are calculated, if average changes of adjacent windows exceed some threshold, this means an edge region exist between this two adjacent windows. Dividing this windows into smaller windows, then calculating the average changes of this small windows , give more accurate edge regions. At the second step, to detect edge pixels, threshold the small edge windows with both their own average value and local variance. With this two step algorihtm , edges can be detected abruptly even at the high noise levels.

Another powerful edge detection algoritm is stokastik gradient method.To detect the presence of n edge at locattion P calculate the horizantal gradient as,

$$g(m,n) = \hat{u}_f(m,n-1) - \hat{u}_b(m,n+1)$$

where $\hat{u}_f(m,n)$ and $\hat{u}_b(m,n)$ are the optimum forward and backward estimates of $u(m,n)$ based on the noisy observations given over some regions W of the left and right half-planes, respectively. Thus $u_f(m,n)$ and $\hat{u}_b(m,n)$ are semicausel estimates. For observations, we can find the best linear mean square semicausel FIR estimate of the form

$$\hat{u}_f(m,n) = \sum_{k,l \in W} \sum a(k,l)v(m-k,n-l) \qquad W=\{(k,l): |k| \le p, 0 \le l \le q\}$$

Where $a(k,l)$ is the Wiener filter weights. The backward semicausel estimate employs the same filter weights but backward. Using the gradient definitions in stochastic model , horizantal mask can be obtained from $u_f(m,n)$ and $\hat{u}_b(m,n)$. Vertical mask would be the 90° counterclockwise rotation of horizantal mask. Using this mask in the directional gradient method, better peformance would be obtained.

In order to detect meaningful edges in noisy pictures, edge detection filters should operate on a global basis and should optimize a givin performance measure. In this chapter, we derive an optimal edge detection filter which produces maximum energy within a resolution interval of specified width in the vicinity of the edge. The product of the resolution and bandwidth of the filter can be set at any desired value. We show that, in the continuous case, the filter transfer function is specified in terms of the prolate spheroidal wave function. In the discrete case, the filter transfer function is specified in terms of the sampled values of the first-order prolate spheroidal wawe function or in terms of the sampled values of an asymptotic approximation of the wave function. Both version can be implamented by FFT transform. By adjusting a single parameter of the filter , it is possible to trade off edge resolution for output signal to noise ratio

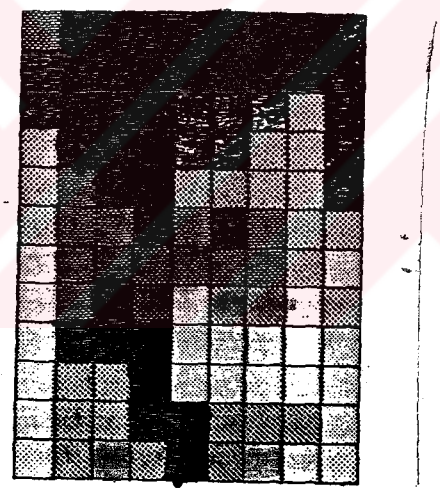
An other method of edge detection is morphological methods. The morfological approach is generally based upon the analysis of two valued image in terms of some predetermined geometric shape known as a structuring element. Essentially, the manner in which the structuring element fits into the image is studied. A simple method of edge detection is to take the difference between an image and its erosion by a small rod shape structuring element. Then choosing an appropriate threshold value, we can get edge image. An improved verson of this method is to take minimum of dilation and erosion residue of the blured image

Edge detection algorithms can be compared in a number of different ways. Quantitavely, the performance in noise of an edge detection operator may be measured as follows. n_o is the number of pixels decleared and n_i is the number of false edge points after adding noise. If no held fixed for the noiseless as well as noisy images, then the edge detection error rate is

$$P_e = \frac{n_i}{n_o}$$

BÖLÜM 1
GİRİŞ

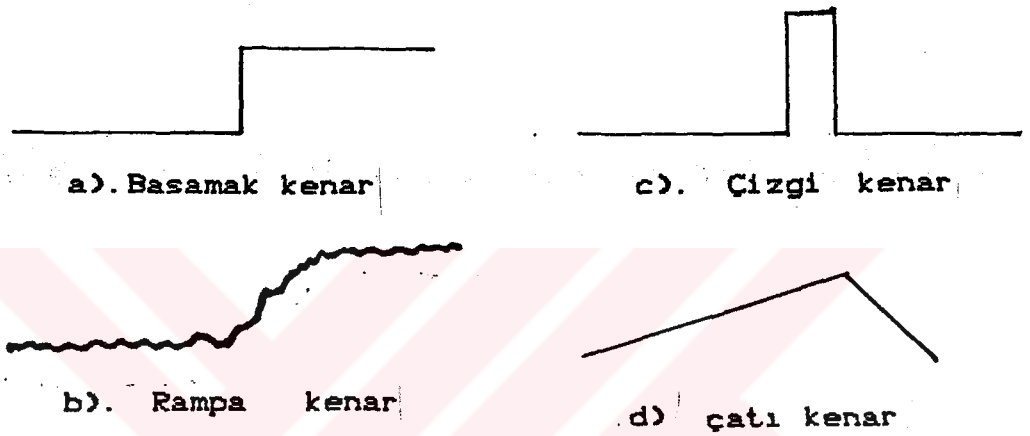
Cisimler farklı renk seviyelerine sahip birçok bölgeden oluşmaktadir. Şekil-1.1 de farkli gri seviyelerine sahip bir görüntü verilmiştir. Cismin gri seviyesi yukarıdan aşağıya belirgin biçimde değişmesine rağmen , görüntünün bir E harfi olduğu kolayca algılanmaktadır. Bunun sebebi , gözün bölgesel kontrasta duyarlı olmasıdır.



Sekil 1.1 değişen gri seviyeli E harfi

Bu bölgesel kontrast farkını analiz edebilmek kenar tanıma algoritmalari ile mümkündür. Gri seviyeleri farklı iki bölge arasındaki geçiş veya sınır bölgesi kenar olarak tanımlanır. Gri seviyelerine göre tanımlanan kenarlar birçok değişik şekilde bulunurlar. En bilinen kenar şekli basamak kenardır (şekil 1.2.a).Basamak kenar, gri seviyeleri homojen ve birbirinden farklı olan iki bölgeyi birbirinden ayıran kenar tipidir. Basamak kenarlar ideal kenar tipidir . Gürültü ve bulanıklık

sebebiyle basamak kenerlar, ikinci tip kenar olan rampa kenarlara dönüştür (şekil 1.2.b). Rampa kenarlar iki bölge arasında gri seviyesi yavaşça değişen bir geçiş bölgesine sahiptir. Diğer bir önemli kenarda cizgi kenardır (şekil 1.2.c). Çizgi kenar her iki yanındaki bölgenin gri seviyesinden farklı gri seviyesine sahiptir. Gri seviyelerin değişimindeki ani değişikliklerde kenar olarak tanımlanabilir. Bu tip kenarlar çatı kenar (şekil 1.2.d) olarak adlandırılır.



Şekil 1.2 Çeşitli kenar fonksiyonları

Kenarlar , görüntülerin en önemli özelliklerinden birisidir. Bir çok alanda kenarların tanınması çok yararlıdır. Örneğin , bir görüntü algılama sisteminde cisimlerin algılanmasında ana basamak , görüntüyü, görüntüdeki farklı cisimlere ait bölgelere ayırmaktır. Kenar tanıma algoritmaları bu segmentasyon işleminin ilk basamağıdır. Ayrıca kenarlar cisimlerin birçok özelliğini içerdiği için sadece kenarları kullanarak etkili görüntü kodlama sistemleri oluşturulabilir.

Bu tezdeki amaç bir dijital görüntüyü giriş verisi olarak kullanıp , verilen görüntüdeki herbir cismin kenarlarını gösteren bir kenar görüntüsü oluşturmaktır. Kenar görüntüsü elde etmek için kullanılan yöntemler 5 bölümde incelenmiştir.

1. Türev yöntemleri

2. Yöresel istatistik yöntemi
3. Filtreleme yöntemi
4. Stokastik gradient yöntemi
5. Morfolojik yöntem

Tezde bu 5 metod incelenecektir. Herbirisi için algoritmalar oluşturulup , kenar görüntüleri elde edilecek, gürültülü ortamlarda metodların performansları karşılaştırılacaktır.

BÖLÜM 2
GÜRÜLTÜ AZALTMA TEKNİKLERİ

Türev alma işleminin gürültüye çok fazla bağlı olması dolayısıyla, kenar tanıma algoritmaları gürültüden dolayı çok fazla yanlış kenar oluşturmakta veya kenarları gerçek yerlerinden kaymış olarak bulmaktadırlar. Bu nedenle görüntüye gürültü azaltma tekniklerinin uygulanması kenar tanıma tekniklerinin performanslarını artırmaktadır. Bu bölümde impuls gürültüleri azaltmada oldukça etkili olan ve görüntüdeki kenar özelliklerini etkilemeyen medyan filtreleri incelenecektir.

2.1 Medyan Filtre

Medyan filtreler impuls gürültüsünü azaltmak için kullanılan doğrusal olmayan filtrelerdir. Gürültüyü azaltırken kenar özelliklerini korumasından dolayı kenar tanıma tekniklerinde kullanılmaktadırlar.

2-B medyan filtrede kxl boyutlarındaki bir pencere görüntünün üzerinde kaydırılarak, pencerenin içindeki piksellerin gri seviyelerinin medyanı işlenen pikselin gri seviyesi olarak bulunur, yani ,

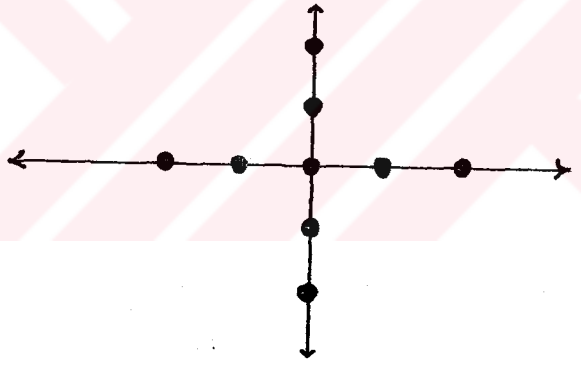
$$f(m,n) = \text{medyan} \{ f(m-k,n-l),\ (k,l) \in W \} \qquad (2.1)$$

burada W, kullanılan pencerenin destek bölgesidir.

Medyan filtrede kullanılan pencerenin boyutları önemlidir. Eğer pencerenin içindeki gürültülü piksel sayısı pencerenin toplam piksel sayısının yarısına eşit veya daha fazla ise filtrenin performansı düşük olacaktır. Bu nedenle optimal filtre boyutunu bulmak için

, değişik boyutlarda medyan filtreler kullanarak , elde edilen sonuçlara göre pencere boyutunu seçmek daha iyidir.

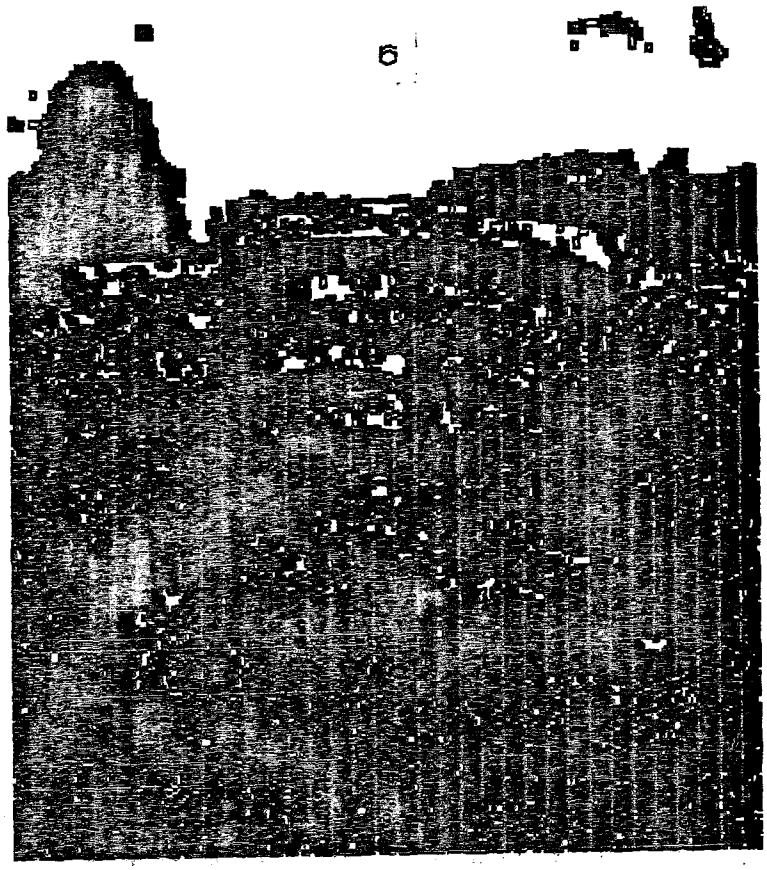
N_w , 2-b medyan filtrenin piksel sayısı ise en büyük $(N_w+1)/2$. inci deger bu pencerenin medyan degeridir. Bu değeri bulabilmek için $3(N_w^2-1)/8$ tane karşılaştırma işlemi yapmak gerekir. Bu sayı 3x3 boyutlu pencereler için 30 , 5x5 boyutlu pencereler içinse 234 'dür. İşlem sayısını azaltmak ve 2-B medyan filtrelerin basamak fonksiyonuna olan etkisini kaldırmak için ayrılabilir medyan filtreler kullanılabilir. Ayrılabilir medyan filtreler 1-B medyan filtrelerin görüntüdeki herbir satır ve sütüna ayrı ayrı uygulanmasıyla elde edilirler. Diğer bir pratik medyan filtre tipi ise (+) tipi medyan filtrelerdir. (Şekil 2.1) Tezde 3x3 ve 5x5 medyan filtreler kullanılmıştır.



Şekil 2.1 (+) tipinde 5x5 medyan filtre

Medyan filtreler impuls gürültüleri için iyi sonuç verirler. Gauss gürültüleri için performansları kötüdür ve görüntüyü bulanıklaştırıp , kenar özelliklerinin kaybolmasına sebep olurlar.

Şekil 2.2 de medyan filtrenin kenar görüntüsüne olan etkisi görülmektedir. Şekil 2.2.a 'da bir araba görüntüsünün Sobel yöntemi ile (eşik değeri=0) elde edilen kenar görüntüsü verilmiştir. Sekil 2.2.b de aynı görüntünün medyan filtre uygulandıktan sonra Sobel yöntemi (eşik değeri=0) ile elde edilen kenar görüntüsü verilmiştir.

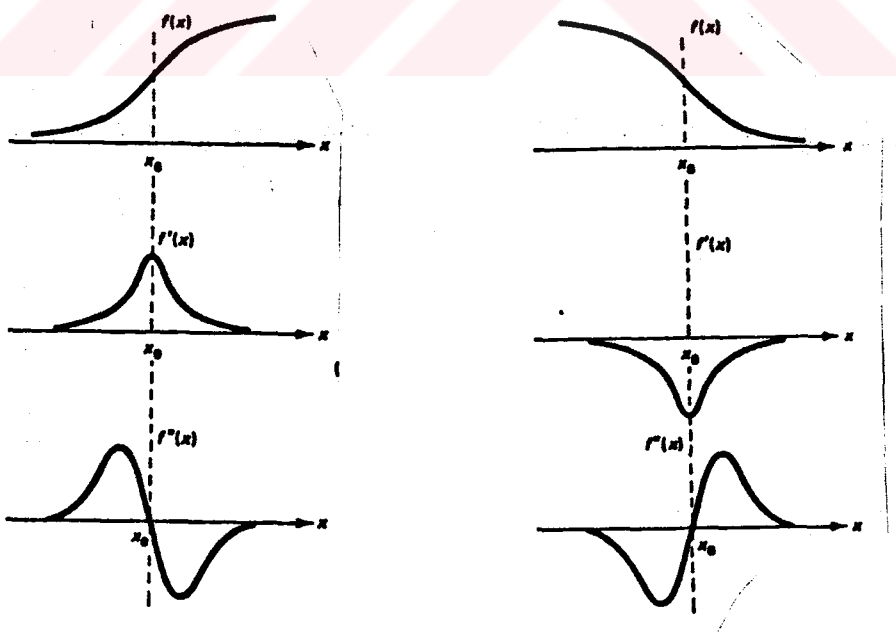


Şekil 2.2 Sobel yöntemi kenar görüntüsü (eşik değeri=0)

Şekil 2.3 5x5 (+) tipi medyan filtrelemeden sonraki Sobel yöntemi kenar görüntüsü (eşik değeri=0)

BÖLÜM 3
TÜREV METODLARI

Kenarları bulmak için uygulanabilecek en verimli yöntemlerden birisi ani gri seviyesi değişimlerini tesbit etmektir. f(i,j) gri seviyesi fonksiyonu için, görüntünün 1. türevi kenar bölgelerinde yöresel maksimum yapar. Bu nedenle görüntünün gradyanını alıp, uygun bir eşik değeri ile sınırlandırırsak kenar görüntüsünü elde etmiş oluruz. Diğer bir yöntemde 2. türevi alarak sıfır geçişlerini tesbit etmektir. Şekil 3-1 'de rampa kenarın 1. ve 2. türevlerinin değişimleri gösterilmektedir. Bu bölümde 1.türevi kullanan gradyent metodu ve 2.türevi kullanan Laplasyan metodu incelenecektir.



Şekil 3.1 Basamak kenarın 1.ve 2.türevleri.

3.1.1 Gradient Metodu

1. türevin 2-B görüntü fonksiyonu $f(i,j)$ ye uygulanması gradyent $\nabla f(i,j)$ 'yi verir. $\nabla f(i,j)$

$$\nabla f(i,j) = \frac{\partial f(i,j)}{\partial x}\cos\theta + \frac{\partial f(i,j)}{\partial y}\sin\theta \qquad (3.1)$$

formulüyle verilir. Burada θ gri seviyelerinin değişim açısıdır. Gradyent, kenar bölgesinde yöresel maksimum yaptığı için , denklem 3-1 'in sıfıra eşitlenmesi kenar büyüklüğünü ve kenar yönünü verir.

$$(\partial/\partial\theta)(\partial/\nabla f(i,j)) = 0 \qquad (3.2)$$

ise , gradyenin büyüklüğü

$$(\nabla f(i,j))_{max} = \sqrt{\frac{\partial f(x,y)^2}{\partial x} + \frac{\partial f(x,y)^2}{\partial y}} \qquad (3.3)$$

ve gradyentin yönü

$$\theta_g = \tan^{-1}(\partial f_y / \partial f_x) \qquad (3.4)$$

ile bulunur.

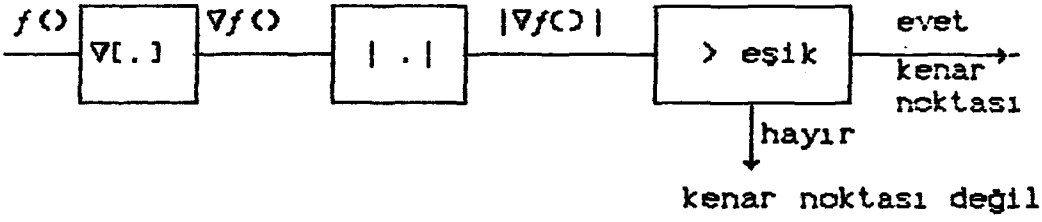
İşlem kolaylığı ve elektronik devre olarak gerçekleştirmesi daha kolay olduğu için gradyentin büyüklüğü genelde

$$|\nabla f(i,j)| = |\partial f_x| + |\partial f_y| \qquad (3.5)$$

formulü ile hesaplanır.

Bu hesaplanan gradyentin büyüklüğü belli bir eşik değerinden büyükse bu pixel kenar noktası olarak tanımlanır. Burada eşik değeri, kenar kabul edilebilecek olan gri seviyesi değişiminin gradyent değeridir. Kenar görüntüsünün performansı seçilen eşik değeri ile çok yakından ilgilidir. Eşik değeri gradyent değerlerinin büyük olanlarını küçük değerlerden ayıracak şekilde seçilmelidir. Uygun eşik değeri, gradyentin histogramına bakılarak belirlenebilir. Genellikle eşik değeri maksimum gradyent değerinin % 5 ile % 10 u olarak

seçilebilir. Kenar görüntüsünde değerinden büyük bütün noktalar kenar noktası olarak tanımlanırsa, oluşan kenar görüntüsü ince çizgiler halinde değil kalın şeritler halinde bulunacaktır. İnce çizgiler elde etmek için kenar görüntüsü kenar inceltme yöntemine sokulurak istenilen inceliğe getirilebilir. Şekil 3-2' de 2-B bir kenar tanıma sisteminin blok diyagramı verilmiştir.



Şekil 3.2 Gradyent metodu blok diyagramı

Gradyent metoduna dayalı kenar tanıma sistemleri 2 değişik şekilde uygulanabilir. Bunlar yönlü kenar operatorü ve yönsüz kenar operatörüdür. $|\nabla f(i,j)|$ fonksiyonunu kullanan sistemlere yönsüz kenar operatörü denir. Çünkü bu sistem herhengi bir doğrultu için ayarlı değildir ve her yön için eşit ağırlıklı sonuçlar verir. Eğer sistem herhangi bir yön için ayarlanmışsa bu sistemlere yönlü kenar tanıma detektörü denir. Örneğin $|\nabla f(i,j)|$ fonksiyonu yerine $|\partial f(i,j)/\partial x|$ 'i kullanan bir sistem dikey doğrultudaki kenarları tanır ve yatay doğrultudaki kenarlara ise cevap vermez.

2-B sayısal fonksiyonlarda türev işlemleri fark denklemleri ile ifade edilebilir. Buna göre $\partial f(i,j)/\partial x$ ve $\partial f(i,j)/\partial y$ türev ifadeleri aşağıdaki fark denklemleri ile değiştirilebilir.[1]

$$\frac{\partial f(i,j)}{\partial x} = [f(i,j)-f(i-1,j)]/T \qquad (3.6.a)$$

veya

$$= [f(i+1,j)-f(i,j)]/T \qquad (3.6.b)$$

veya

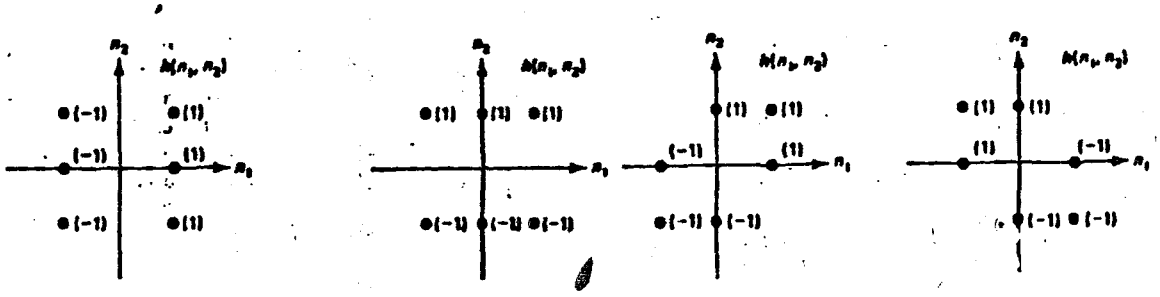
$$= [f(i+1,j)-f(i-1,j)]/2T \qquad (3.6.c)$$

Bu ifadelerin daha iyi yaklaşımları için aşağıdaki ifadeler kullanılabilir.

$$\frac{\partial f(i,j)}{\partial x} = [f(i+1,j+1) - f(i-1,j+1) + f(i+1,j) - f(i-1,j) + f(i+1,j-1) + f(i-1,j-1)] \quad (3.7.a)$$

$$\frac{\partial f(i,j)}{\partial y} = [f(i+1,j+1) - f(i+1,j-1) + f(i,j+1) - f(i,j-1) + f(i-1,j+1) - f(i-1,j-1)] \quad (3.7.b)$$

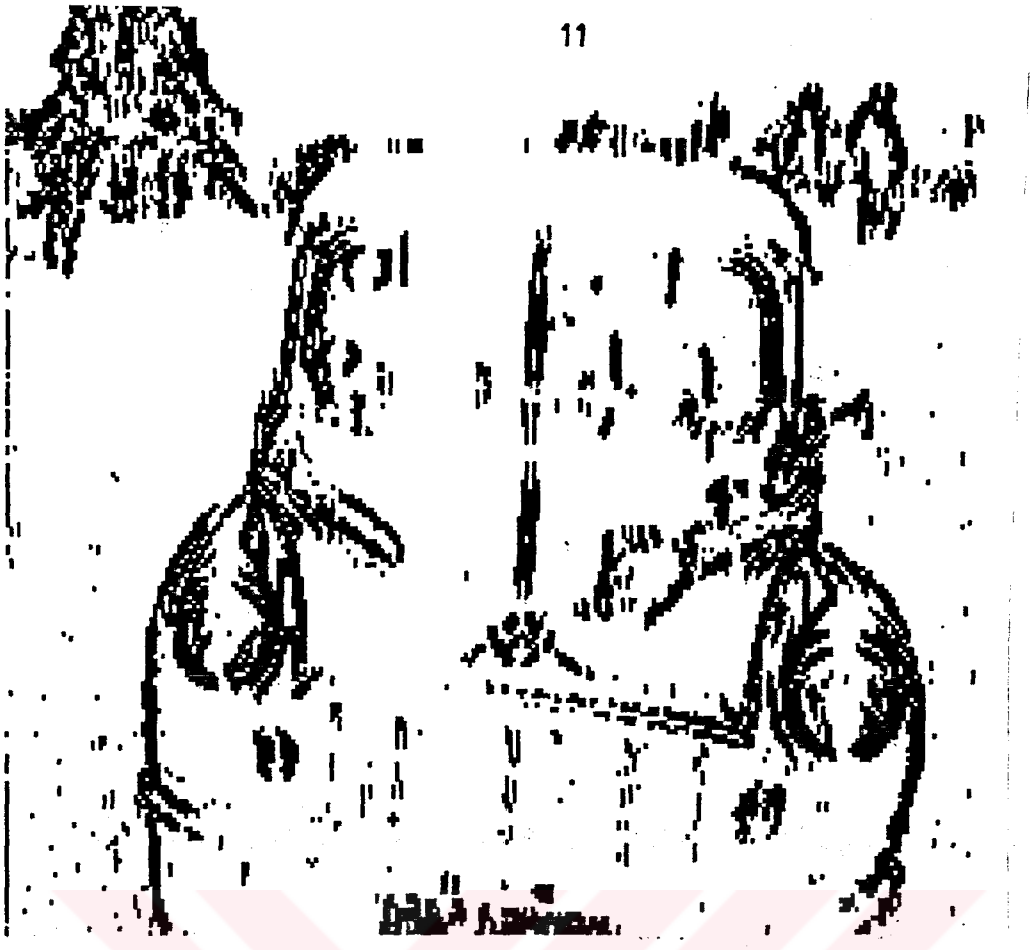
Denklemlerde T parametresi iptal edilmiştir. Çünkü eşik değeri T oranında daha büyük verilerek aynı sonuç daha az matamatiksel işlemle elde edilmiş olur. Denklem 3-7.a ve3-7.b deki fark denklemlerinin $f(i,j)$ fonksiyonunun impuls cevabı $h(i,j)$ olan bir filtre ile konvolüsyonuna eşit olduğu görülmektedir. Buradaki $h(i,j)$ impuls cevaplı filtreler kullanılan yönlü ve yönsüz kenar operatörleridir. Bu filtrelere maske ve konvolüsyon işleminede maskeleme denilmektedir. Şekil 3-3-a 'daki $h(i,j)$ filtresinden $\partial f(i,j)/\partial x$ 'i elde ederek dikey doğrultudaki kenarları bulmak için ve şekil 3-3-b deki $h(i,j)$ filtreside $\partial f(i,j)/\partial y$ 'i elde ederek yatay doğrultudaki kenarları bulmak için kullanılabilir. Şekil 3-3-c ve 3-3-d $h(i,j)$ filtreleri $\pm 45^{\circ}$ doğrultusundaki kenarları bulmak için kullanılabilir.



a) Dikey b) Yatay c) d) Ortogonal

Şekil 3.3 Yönlü Kenar Maskeleri

Şekil 3.4 de yönlü kenar maskeleri ile elde edilmiş kenar görüntüleri verilmiştir. Görüldüğü maskeler sadece



Şekil 3.4.a Dikey kenar operatörü, eşik=3

Şekil 3.4.b Yatay kenar operatörü, eşik=3

kendi doğrultuları için kenar tanımakta diğer doğrultulardaki kenarları ise tanıyamamaktadırlar.

Yönsüz kenar operatörleride fark denklemleri yaklaşımıyla elde edilebilir. Yönsüz kenar operatörlerinde hem x hem de y doğrultusunda iki ayrı filtre ile konvolusyon işlemi uygulanır ve kareleri toplamının karekökü alınarak gradientin mutlak değeri bulunur.

$$|\nabla f(i,j)| = \sqrt{\partial f_x(i,j) + \partial f_y(i,j)} \qquad (3.8)$$

burada

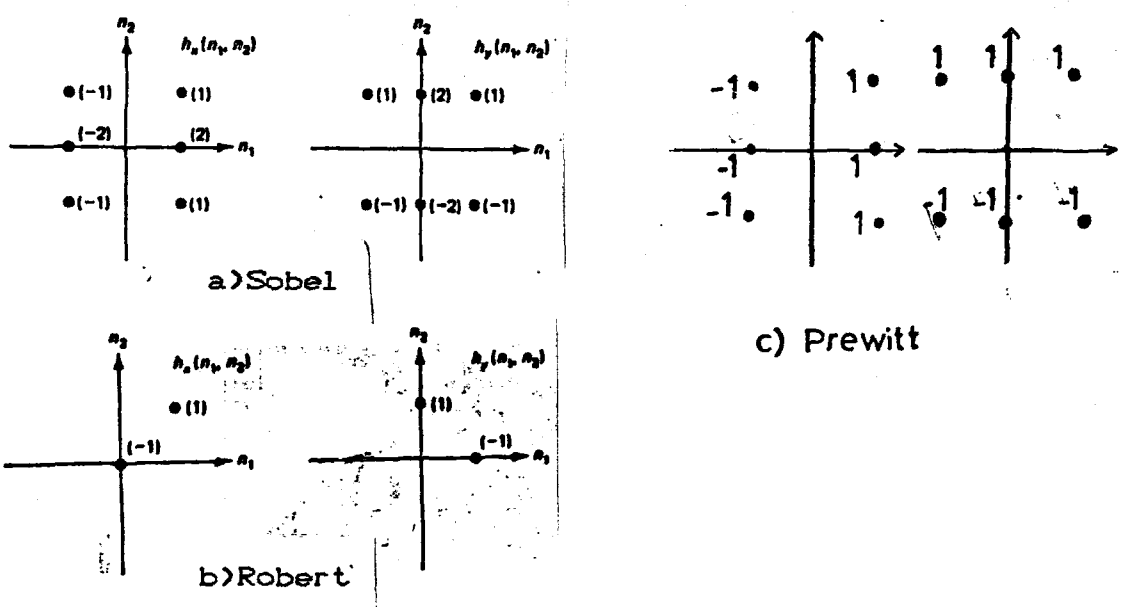
$$\partial f_x(i,j) = f(i,j) * h_x(i,j) \qquad (3.9.a)$$

ve

$$\partial f_y(i,j) = f(i,j) * h_y(i,j) \qquad (3.9.b)$$

burada $h_x(i,j)$ ve $h_y(i,j)$yönsüz kenar tanıma operatörlerininx ve y doğrultusundaki maskeleridir.

Günümüze kadar birçok yönsüz kenar tanıma maskeleri geliştirilmiştir. Bunlardan ençok bilinenleri Sobel, Robert ve Prewitt kenar tanıma maskeleri şekil 3-5' de gösterilmiştir.



Şekil 3.5 Yönsüz Kenar Maskeleri

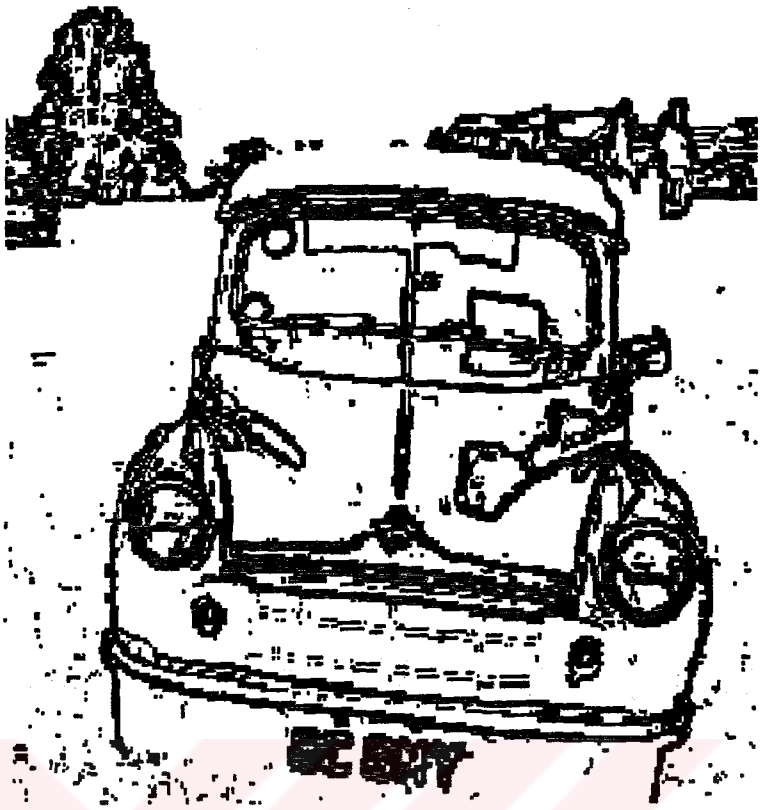
Şekil 3.6'da bu maskelerle elde edilmiş kenar görüntüleri verilmiştir. Bu maskeler her yöndeki kenarları tanımaya çalışmaktadırlar. Diagonaldeki görüntüler için Robert maskesinin daha etkili olduğu görülmektedir. Fakat resimleri bütün olarak ele aldığımızda Sobel maskesinin daha etkili kenar görüntüleri oluşturduğu görülmektedir.

3.1.2 Kompas Kenar Oparatörü

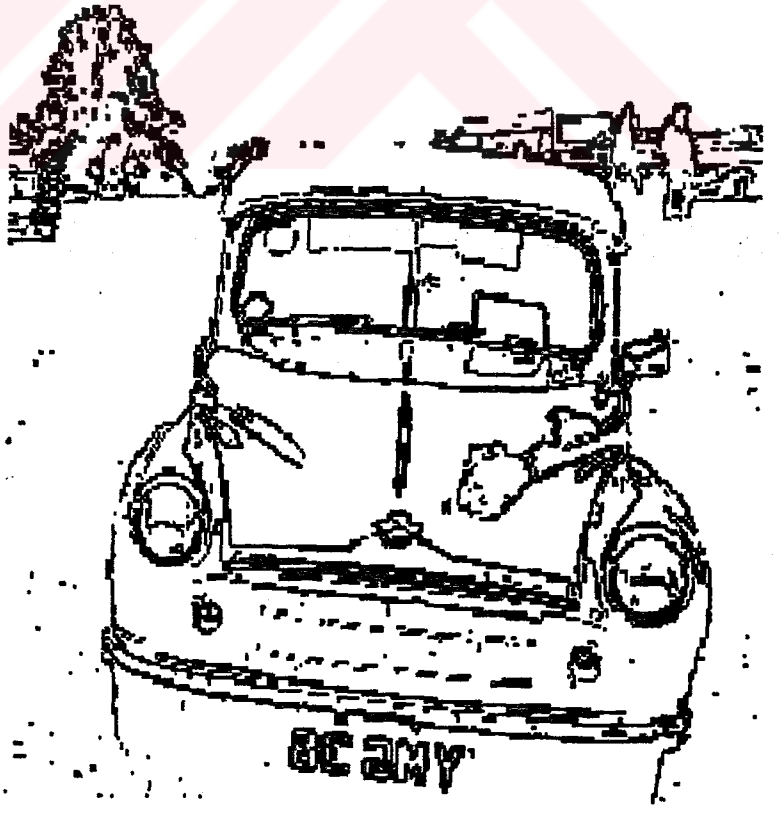
Önceki bölümde anlatılan yöntemlerde dikey ,yatay veya Robert metodunda olduğu gibi ±45° açı yapan kenarlar tanınabiliyordu. Gradient metodunun biraz değiştirilmiş hali olan kompas maskesi 45°nin katlari olan bütün açılarda kenarları tanımaya yarar. Bir periyotta 8 tane 45° olduğu için , herbirisi bir öncekinin 45° döndürülmüş halı olan 8 ayrı maske ile görüntü konvolüsyona sokulur ve görüntüdeki herbir pixel için bu 8 ayrı gradyentten en büyük olanı seçilir. Bu çıkış gradyentinin bir eşik değeri ile sınırlandırılması ile kenar görüntüsü elde edilir.[2]

Gradyent kenar tanıma algoritmalarında olduğu gibi, genel kullanımda birçok kompas maskeleri vardır. En bilinenlerinden Prewitt, Kirsh maskeleri şekil 3.7 'de verilmiştir. Bu maskelerin herbirisi bir öncekinin 45°döndürülmüş halidir. 8 maskeden 4 tanesinin diğer 4 tanesinin simetriği olması ve eşik değeri alınırken mutlak değer işlemi uygulanması nedeniyle,kompas maskelerini kullanırken görüntüyü ilk 4 maskeyle konvolüsyona sokmak ve bu dört tanesini kullanarak masksimum gradyent değerini bulmak kenar görüntüsünü elde etmek için yeterli olacaktır.

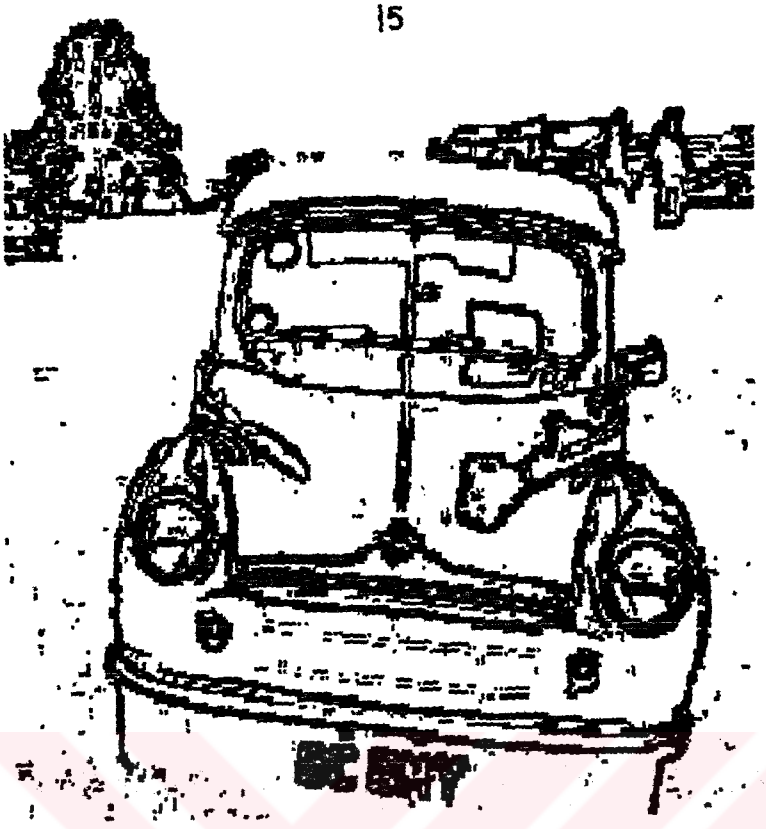
Şekil 3.8 de kompas maskesi ile elde edilmiş kenar görüntüleri verilmiştir. Her yön için ayrı ayrı kenar tanıma işlemi uygulandığı için, beklenildiği gibi daha



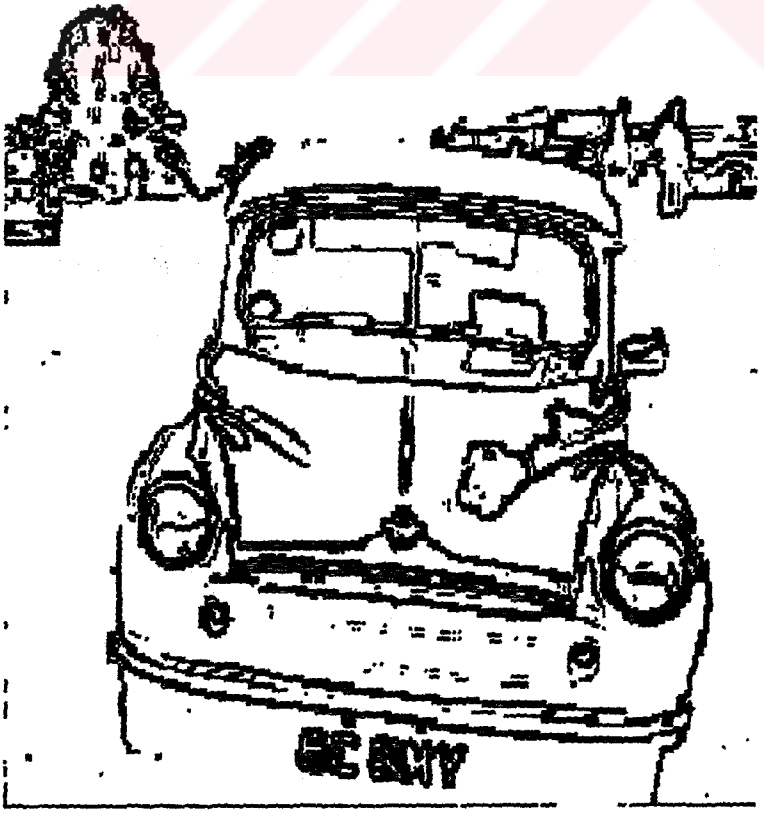
Şekil 3.6.a Sobel kenar operatörü, eşik=11



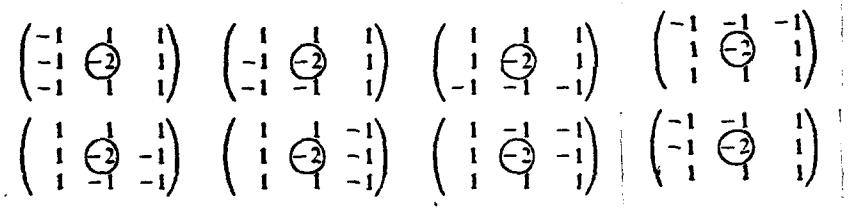
Şekil 3.6.b Robert kenar operatörü, eşik=5



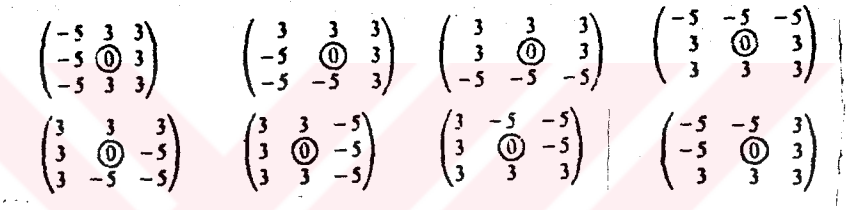
Şekil 3.6.b Prewitt kenar operatörü, eşik=8



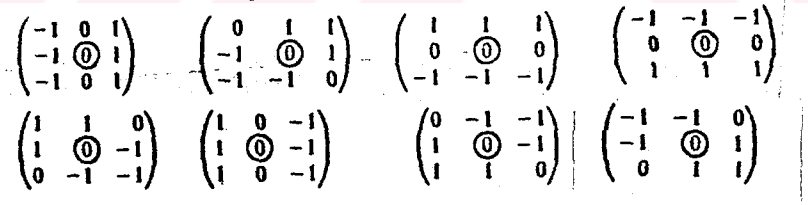
Şekil 3.8 Kompas kenar operatörü, eşik=9



a) Prewit Maskesi



b) Kirsh Maskesi



c) 8 Adet 3 Seviyeli Maske

Şekil 3.7 Kompass maskeleri

net kenar görüntüleri elde edilmiştir.

3.2.1 Laplasyan Kenar Tanıma

Diğer bir kenar tanıma yöntemide 2.türevi kullanmaktır. Şekil 3-1.c görüldüğü gibi kenar bölgelerinde 2.türev sıfırdır. Bu nedenle $f(i,j)$ görüntü fonksiyonunun 2. türevini alıp sıfır geçiş noktalarını tesbit edersek kenar görüntüsünü elde etmiş oluruz.

2-B görüntü fonksiyonu $f(i,j)$ in 2. türevine Laplasyan denir ve $\nabla^2 f(i,j)$ ile gösterilir. Laplasyan $\nabla^2 f(i,j)$ aşağıdaki formülle bulunur.

$$\nabla^2 f(i,j) = \nabla(\nabla f(i,j)) = \frac{\partial^2 f(i,j)}{\partial x^2} + \frac{\partial^2 f(i,j)}{\partial y^2} \qquad (3.10)$$

Gradyent operatörlerinde olduğu gibi Laplasyan'da fark denklemleri ile ifade edilebilir. 2. dereceden fark denklemiaşağıdaki şekilde ifade edilebilir.

$$\frac{\partial f(i,j)}{\partial x} = f_x(i,j) = f(i+1,j) - f(i,j) \qquad (3.11)$$

Bu sefer $f_x(i,j)$ in türevini ters yönde fark denklemi ile ifade edersek

$$\frac{\partial^2 f(i,j)}{\partial x^2} = f_{xx}(i,j) = f_x(i,j) - f_x(i-1,j) \qquad (3.12)$$

Denklem 3-11'i denklem 3-12 'ye yerleştirirsek

$$f_{xx}(i,j) = f(i+1,j) - 2f(i,j) + f(i-1,j) \qquad (3.13.a)$$

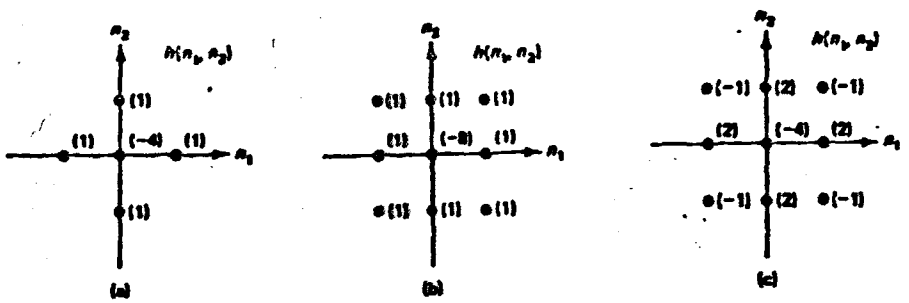
i elde ederiz. Aynı şekilde

$$f_{yy}(i,j) = f(i,j+1) - 2f(i,j) + f(i,j-1) \qquad (3.13.b)$$

olarak elde edilir. Denklem 3-13.a ve denklem 3-13.b yi birleştirirsek Laplasyanı

$$\nabla^2 f(i,j) = f(i,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j) \qquad (3.14)$$

Elde edilen $\nabla^2 f(i,j)$, $f(i,j)$ fonksiyonunun $h(i,j)$ impuls cevaplı filtrenin konvolusyonuna eşittir. $h(i,j)$ filtre impuls cevabı için yukarıda elde edilen fark denklemi kullanılabileceği gibi şekil 3.9..b ve 3.9 .c 'deki impuls cevaplarıda kullanılabilir.



Şekil 3.9. Laplasyan Kenar Maskeleri

Laplasyan kenar tanıma metodunda, $f(i,j)$ görüntü fonksiyonu herhangi bir laplasyan maskesi ile konvolüsyona sokulduktan sonra ,mutlak değeri alınır ve belli bir eşik değeri ile sınırlandırılırsa kenar görüntüsü elde edilir.

Gradyent metodu ile aynı işlemler yapılmasına rağmen elde edilen kenar görüntüsünde bazı farklar vardır. Şekil 3.1 'den de görüldüğü gibi mutlak değer işlemi laplasyan metodunda çift kenar oluşturur. Diğer bir farkta , sıfır geçişleri cisimlerin sınırlarını gösterdiği için, kenar görüntüsü sürekli çizgiler halinde olur. Bu yüzden kenar inceltme metodlarına ihtiyaç yoktur. Fakat $f(i,j)$ fonksiyonundaki çok küçük oynamalar bile bir sıfır geçişi oluşturacakları için çok fazla yanlış kenar noktası oluşacaktır.

Bu yanlış kenar noktalarını yok etmek için uygulanabileçek bir yöntem varyans yöntemidir. Varyans yönteminde sıfır geçişi noktalarının kenar noktası kabul edilebilmesi için , o noktadaki yöresel varyansın yeterince büyük olması gerekmektedir. Yöresel varyans aşağıdaki formülle bulunabilir.

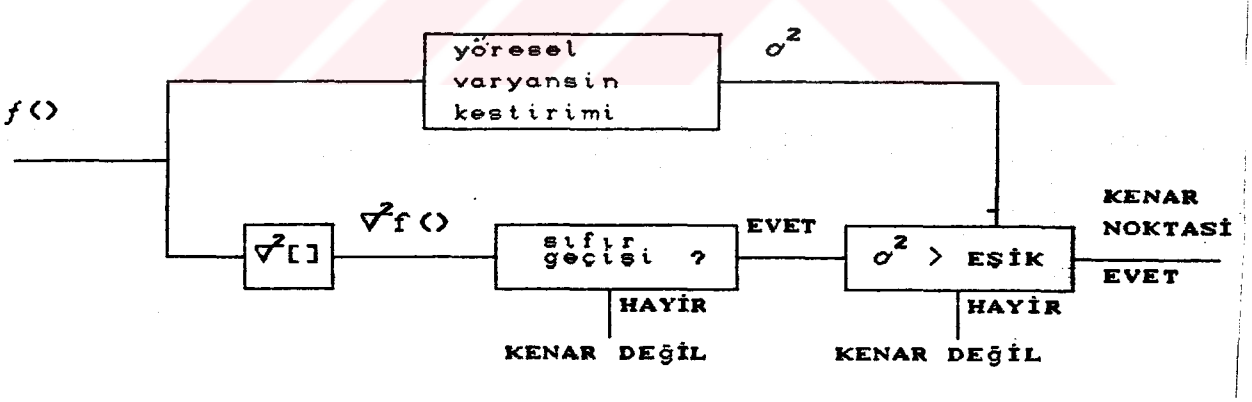
$$\sigma_f^2(i,j) = \frac{1}{(2M+1)^2} \sum_{k=i-M}^{i+M} \sum_{l=j-M}^{j+M} [f(i,j)-m(k,l)]^2 \qquad (3.15.a)$$

burada

$$m(i,j) = \frac{1}{(2M+1)} \sum_{k=i-M}^{i+M} \sum_{l=j-M}^{j+M} f(k,l) \qquad (3.15.b)$$

Tezde M değeri 1 olarak seçilmiştir. Büyük değerler için destek bölgesi çok büyümektedir. Varyans testi ile yanlış kenarların sayısında önemli bir düşme görülmektedir. Şekil 3.10 de normal laplasyan metodu ile elde edilmiş kenar görüntüsü verilmiştir. Şekil 3.11 da varyans testi ile hatalı tanınan kenar noktalarındaki azalma belirgin olarak görülmektadir.

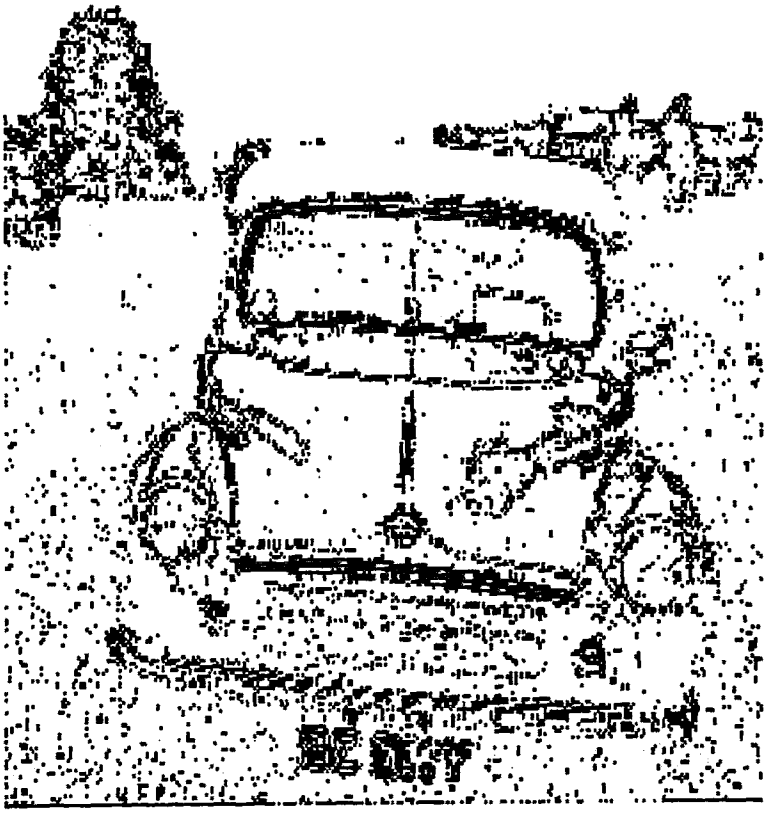
Şekil 3.12 de Laplasyan metodu kenar tanıma metodunun blok diyagramı gösterilmektedir.



Şekil 3.12 Laplasyan metodu blok diyagramı

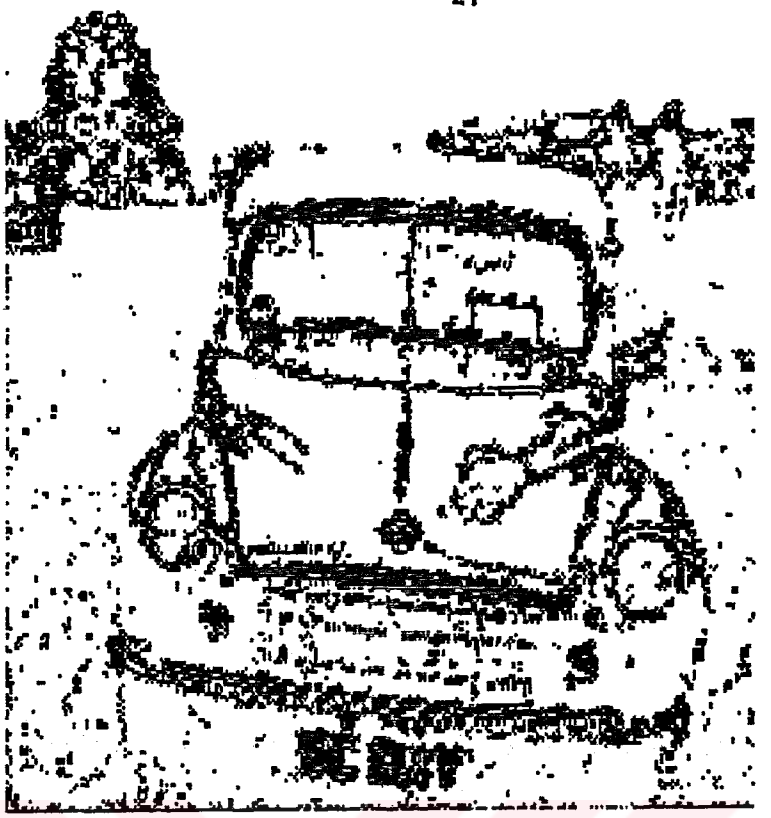
3.2.2 Laplasyan-Gauss Metodu

Laplasyan-Gauss metodu ilk olarak ilk olarak Marr ve Hilbert tarafından önerilmiştir. Marr ve Hilbert görüntüyü değişik kesim frekanslarında sınırlayıp ve



Şekil 3.10.a Laplasyan operatörü, eşik=5

Şekil 3.10.b Laplasyan operatörü, eşik=5



Şekil 3.11.a Varyans testli Laplasyan metodu, varyans=2 eşik=3

Şekil 3.11.b Varyans testli Laplasyan metodu, varyans=2 eşik=3

herbir sınırlandırılmış görüntü için ayrı kenar tanıma metodları uygulayarak değişik kenar görüntüleri elde etmeyi önermişlerdir. Eğer bir kenar bu değişik çözünürlüklere ait kenar görüntüleri setinde bulunuyorsa , bu kenar bir gri seviyesi değişimini temsil eder.

Görüntüyü değişik kesim frekanslarında sınırlandırmak için Gauss alçak geçirgen filtresi kullanılmıştır .Gauss filtresi impuls cevabı

$$h(x,y) = e^{-(x^2+y^2)/(2\pi\sigma^2)} \qquad (3.16)$$

ile ifade edilir. Burada σ kesim frekansını belirler.

Gauss filtresinden geçirilmiş görüntünün kenarları için Laplasyan kenar tanıma metodu uygulanmıştır Laplasyan maskesini bu filtrelenmiş görüntüye uygular ve $\nabla^2[.]$ ve konvolüsyon (*) işlemlerinin doğrusallık özelliklerini kullanırsak

$$\nabla^2(f(x,y)*h(x,y)) = f(x,y) * [\nabla^2 h(x,y)] \qquad (3.17)$$

elde edilir. Bu denklemdeki Gauss fonksiyonu $h(x,y)$ için $\nabla^2 h(x,y)$ aşağıda verilmiştir.

$$\nabla^2 h(x,y) = \frac{e^{-(x^2+y^2)/(2\pi\sigma^2)}}{(\pi\sigma^2)^2}(x^2+y^2-2\pi\sigma^2) \qquad (3.18)$$

Denklem 3.18deki formüle Laplasyan-Gauss maskesi denir ve LoG ile gösterilir.

LoG metodunda , görüntü LoG maskesi ile konvolüsyona sokulur ve sıfır geçiş noktaları bulunur. $f(x,y)$ sınırlandırıldığı için , Laplasyan metodundaki gürültü hassasiyeti kalmaz.

LoG filtresinin hassasiyeti merkezi etki alanının büyüklüğü ile ilgilidir. Merkezi etki alanı ,w, yaklaşık 3σ dir. LoG oparatörünün büyüklüğü ,s, ise $2\sqrt{2}w$

olmaktadir. Bu bölgenin dışında ∇ h yaklaşık sıfıra eşittir. w ne kadar küçük olursa loG filtresi o kadar hassas olur. Buna karşın gürültü tolaransı o kadar kötü olur. Genişliği w dan küçük olan kenarlar hassas olarak tesbit edilebilirler. Buna karşın genişliği w dan daha büyük olan kenarlar gerçek yerlerinden kaymış olarak tesbit edilirler. Rampa kenarlarin tesbit edilebilmesi için genisliklerinin s den küçük olması gerekmektedir. Çatı kenarları içinse ,kenarın genişliği w dan azsa , kenar (w-d)/2 kadar kaymış olarak tesbit edilir.

LoG operatörünün en önemli dezavantajı matemetiksel işlemlerinin çokluğudur. Çünkü herbir pixel için MxM (M ,filtrenin büyüklüğü) işlem gerekmektedir.Fakat bu sorun LoG filtresinin ayrılabilir olması özelliğinden dolayı azalmiştir.LoG filtresi iki ayrılabilir 2-B filtre cinsinden ifade edilebilir.[3]

$$\nabla^2 h(x,y) = h_{12}(x,y) + h_{21}(x,y) \qquad (3.19.a)$$

$$h_{12}(x,y) = h_1(x)h_2(y) \qquad (3.19.b)$$

$$h_{21}(x,y) = h_2(x)h_1(y) \qquad (3.19.c)$$

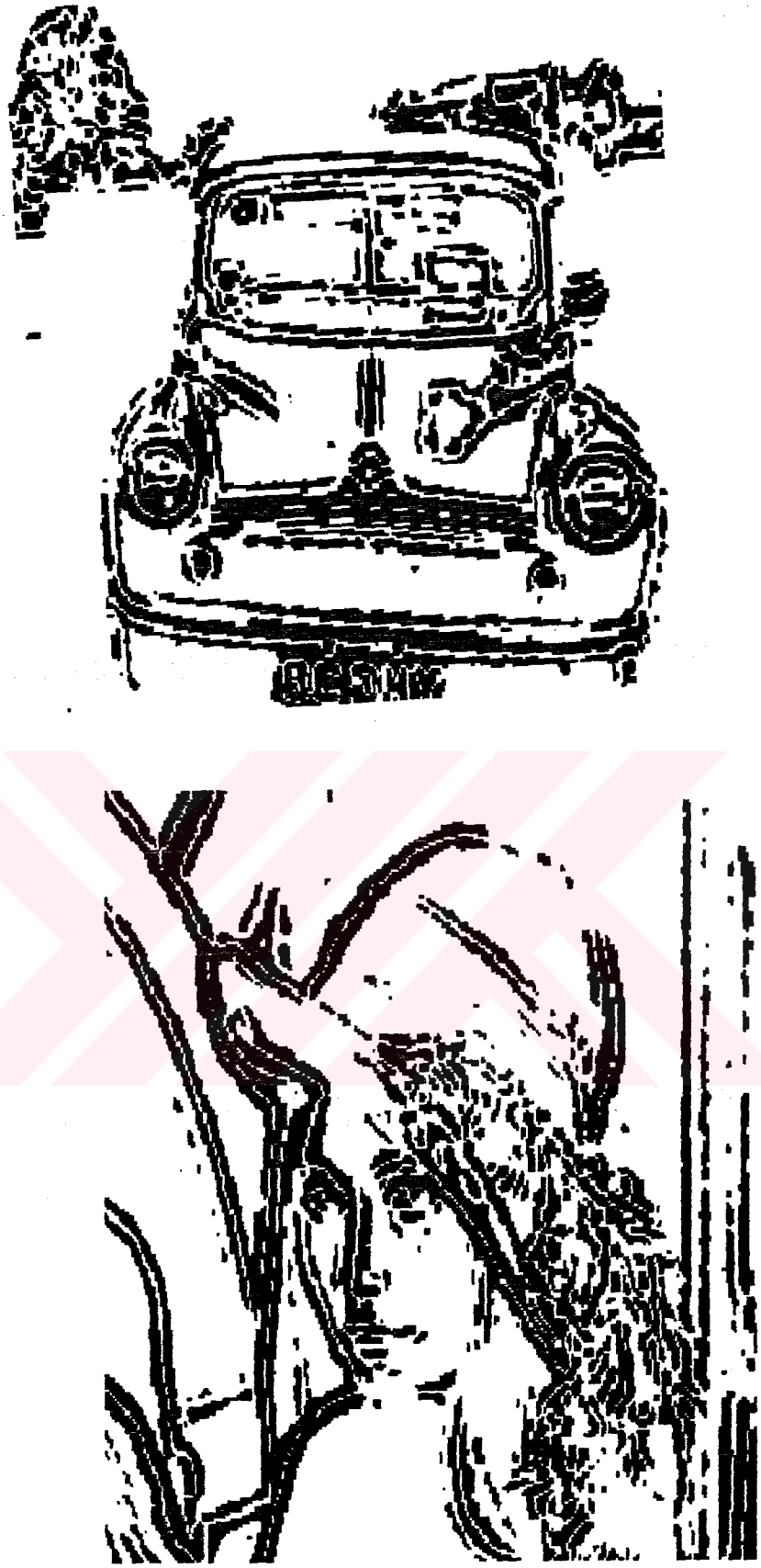
burada

$$h_1(\zeta) = \sqrt{K}\left[1 - \frac{\zeta}{\sigma^2}\right]\exp\left[\frac{\zeta^2}{2\sigma^2}\right] \qquad (3.20.a)$$

$$h_2(\zeta) = \sqrt{K}\exp\left[\frac{-\zeta^2}{2\sigma^2}\right] \qquad (3.20.b)$$

burada K sabit bir çarpandır.

Şekil 3.13 de LoG filtresi kullanılarak elde edilen test kenar görüntüleri verilmiştir. LoG filtresinin alçak geçirgen filtre olması özelliği dolayısıyla kenarlardaki keskinlik azalmış daha yumuşak geçişli kenar görüntüleri elde edilmiştir. Gürültü yüksek frekanslı olduğu için gürültüyü süzer. Bu nedenle gürültüye karşı oldukça etkili bir yöntemdir.



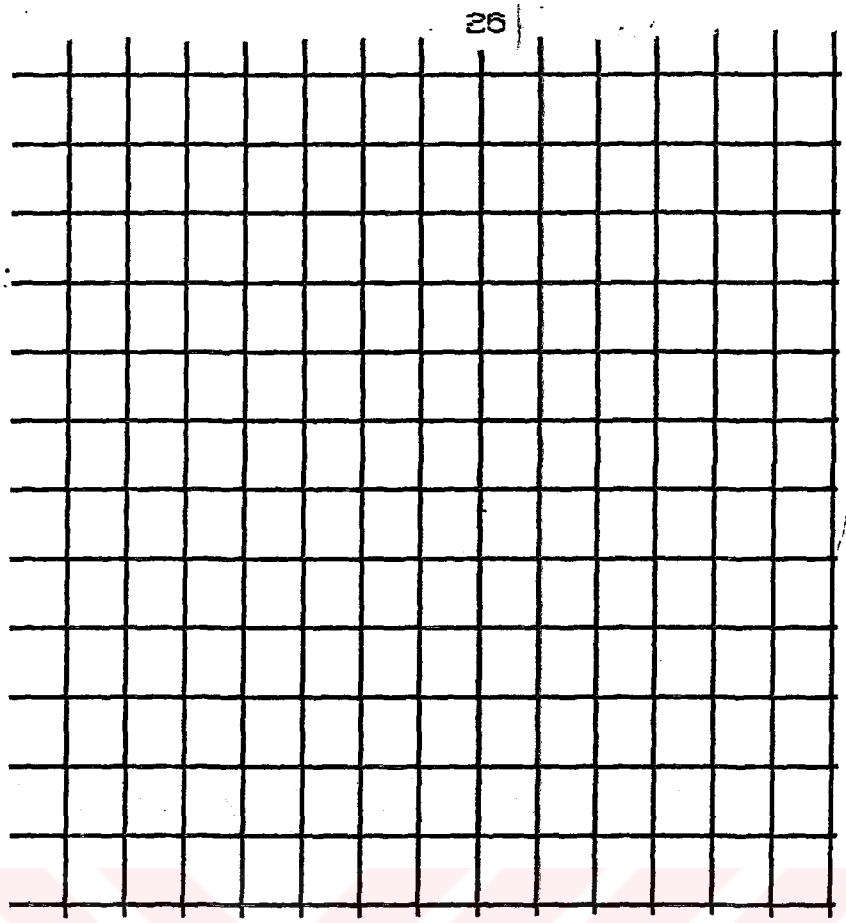
Şekil 3.13 LoG filtre görüntüsü, varyans=1, eşik=2

3.3. Sonuç

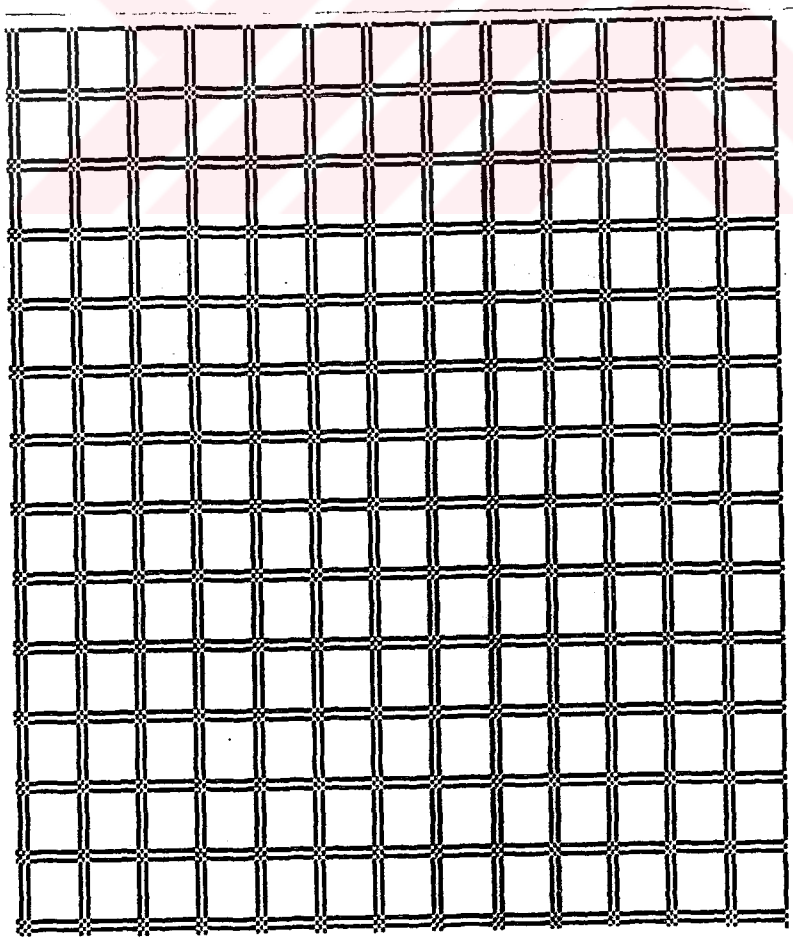
Bu bölümde geleneksel kenar tanıma metodları olan türev metodları incelendi. Bu metodlarda ilk dikkat edilecek nokta, kenarların yerlerinden kaymış olarak tesbit etmesidir. Bu durum şekil 3.14 ve 3.15 de ızgara test resminde görülmektedir.

Diğer bir noktada bu metodların gürültüye karşı hassas olmalarıdır. Çünkü iki gri seviyesi arasındaki farkı alarak kenarları tesbit eden bu metodlar, gürültüye karşı çok hassasdırlar. Ayrıca görüntüdeki bulanıklık diğer bir problemdir. Kenarların yavaşça ve geniş bir bölgede değişmeleri, kenarların tesbit edilememesine veya kaymasına sebep olmaktadır.

Özelikle Laplasyan metodu gürültüye karşı çok hassasdır. Bu yüzden pratik uygulamalarda kullanılmamaktadır. Fakat Laplasyan metodunun filtreler ile kullanılması etkili sonuçlar vermektedir. Bu bölümde incelenen LoG filtreside gürültüye karşı etkilidir. Alçak geçirgen filtre özelliği dolayısıyla gürültüyü süzmektedir. Fakat, destek bölgesi büyüdükçe, işlem sayısı çok artmaktadır.



Şekil 3.14 Izgara test resmi



Şekil 3.15 Izgara test resmi kenar görüntüsü

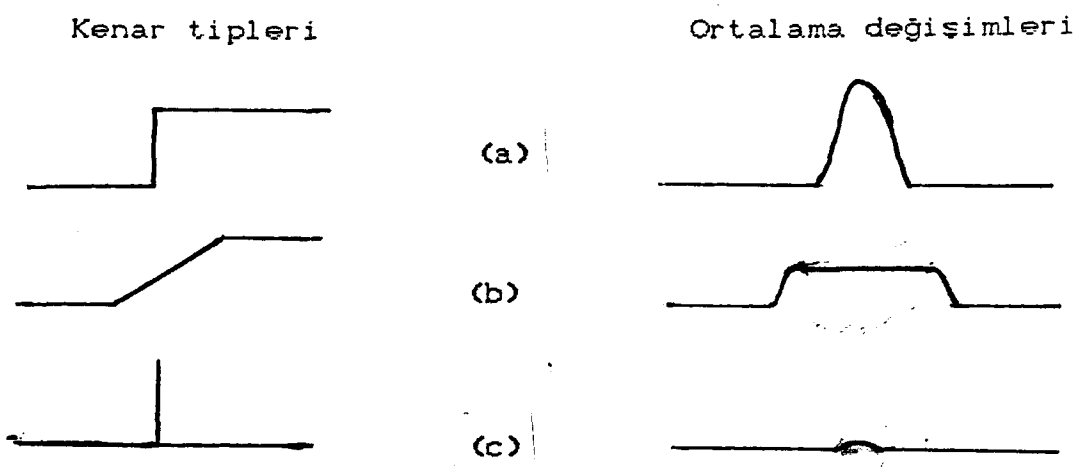
BÖLÜM 4
YÖRESEL İSTATİSTİK METODU

Bu bölümde yöresel ortalama ve varyans değişimlerini bularak kenarları tesbit eden bir yöntem inceleyeceğiz. Algoritma iki aşamadır. Birinci aşamada, görüntüyü küçük pencerelere bölerek, pencerelerin ortalamalarındaki değişmeler incelenir ve kenar içeren pencereler bulunur [4]. İkinci aşamada, içinde kenar içeren pencereler , yöresel eşik değerleri ile eşiklenir ve bu noktaların yöresel varyansları hesaplanarak kenar noktaları bulunur.[5]

4.1 Pencereleme Basamağı

Bu basamakta gri seviyelerinde meydana gelen değişmeler, kabaca belirlenir. Bunu belirlemenin en kolay yolu gri seviyeleri ortalamalarında meydana gelen değişimleri bulmaktır. Görüntü MxM boyutlarında küçük pencerelere bölünür. Herbir pencerenin gri seviyesi ortalamaları hesaplanır ve gri seviyeleri arasındaki farklar hesaplanır. Gri seviyesi farkı belli bir eşik değerini geçen pencerelerde kenar olduğu kabul edilir. Şekil 4.1 de bazı kenar tiplerinde yöresel ortalamaların değişimini göstermektedir.

Bu yöntemde pencerelerin büyüklüğü önemlidir. Pencerenin büyüklüğü , içinden iki tane kenar geçmiyecek kadar küçük ve aynı zamanda da rampa kenarları da içine alacak kadar büyük olmalıdır. Ayrıca gürültü faktörüde diğer bir etkendir. Sistemde toplamsal beyaz gürültü olduğunu kabul edersek , beyaz gürültü süreçlerinin

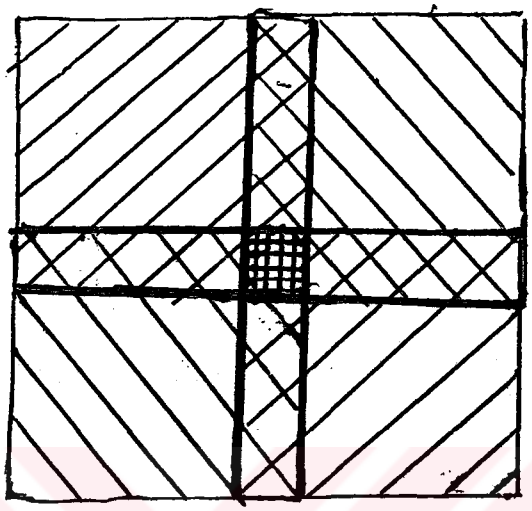


Şekil 4.1 Bazı kenar tiplerinin yöresel ortalama değişimleri

ortalaması sıfır olduğu için, pencereyi ne kadar büyük seçersek, gürültünün ortalama değişimlerine olan etkisi, minimuma iner. Böylece gürültü bağımlılığı düşük bir bir yöntem elde etmiş oluruz. Bu şekilde çok düşük SNR oranlarında bile kenar bölgeleri kabaca belirlenebilir.

Bütün kenarları tesbit etmek ve her bir pencerede sadece tek bir kenar bulunmasını sağlamak için pencereleme basamağı iki aşamada yapılmıştır. Birinci aşamada,belli bölgeleri komşu pencerelerle çakışacak şekilde büyük pencereler seçilmektedir.Bu büyük pencerelerin gri seviyeleri ortalamaları komşu pencerelerin gri seviyeleri ile karşılaştırılır.Eğer gri seviyesi farkı belli bir eşik değerini geçerse bu pencerede kenar bulunmaktadır ve ikinci aşama uygulanır. İkinci aşamada, büyük pencerede alt pencerelere bölünerek, küçük pencerlerin ortalama farkları hesaplanır. Böylece kenarlar küçük pencerelere indirgenmiş olur. Tezde, ilk aşamada 5x5 boyutlarında pencereler seçildi. Pencerelerin 1,2,4,5. sütunları yanlarındaki pencerelerle 1,2,4,5. satırlarıda alt ve üstündeki pencerelerle çakışmaktadır. (Şekil 4.2.). Herbir büyük pencerede 3x3 boyutlarında alt pencerelere bölünmüşlerdir. Bu alt pencerelerinde 1. ve 3.satır ve

sütunları çevrelerindeki alt pencerelerle çakışmaktadır . Alt pencere ortalama farklarıda belli bir eşik değerini geçerse , o alt pencere bir kenar penceresi olarak tesbit edilir ve kenarın tam yerini belirlemek için bir sonraki bölümdeki işlemler uygulanır.



Şekil 4.2 Görüntünün pencerelere bölünmesi

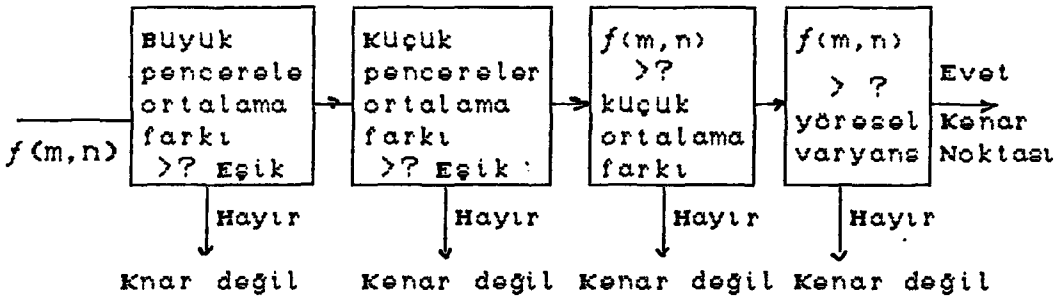
4.2 Kenar Piksellerinin Bulunması

Kenar bölgesi tesbit edilen pencerelerde , kenar noktalarını tesbit edebilmek için yeni işlemler gerekmektedir. Çünkü alt penceredeki her piksel kenar değildir ve o şekilde bırakılırsa çok kalın kenar görüntüsü elde edilir. Bunun için iki işlem uygulanır. Eşik değeri ve varyans testi.

Alt pencerede sadece iki gri seviyesinin bulunduğunu varsayabiliriz. Kenar bölgesi olduğu için bu iki gri seviyesi kenarın iki yanını aittir. Bu nedenle alt pencerenin gri seviyesi ortalaması, bu iki gri seviyesinin ortasında kalır ve eşik değeri olarak seçilirse kenar geçişini tesbit edebilir. Eşik değeri kenarın üst tarafındaki bütün pikselleri kenar olarak tanımlar . Bu nedenle gerçek kenar noktalarını bulabilmek için yöresel varyans testi uygulanır (denklem 3.15). Eşik varyans değerini geçen görüntü noktaları

kenar olarak tanımlanır.

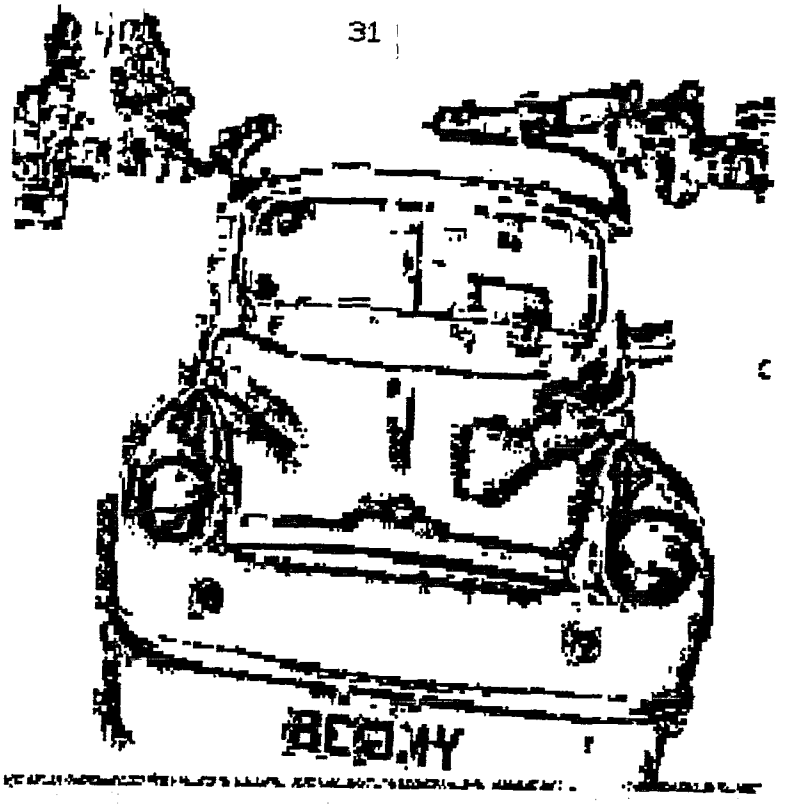
Bu sistemin blok diyagramı aşağıda gösterilmektedir.



Şekil 4.3 Yöresel istatistik metodu blok diyagramı.

4.3 Sonuç

Şekil 4.4 de bu yöntemle elde edilen kenar görüntüleri verilmiştir. Yöresel ortalamalara göre kenarlar bulunduğu için kenar görüntüleri gürültüsüz görüntüler için 3.bölümdeki türev metodlarına göre daha kötüdürler. Fakat gürültülü görüntüler için yüksek gürültü oranlarında da kenar görüntüsü kabaca elde edilebilmektedir. Şekil 4.5 de gürültülü araba görüntüsünün kenar görüntüsü verilmiştir.



Şekil 4.3.a. Yöresel istatistik metodu, varyans=3, büyük pencere ortalama farkı=3, küçük pencere ortalama farkı=2

Şekil 4.3.b. Yöresel istatistik metodu, varyans=3, büyük pencere ortalama farkı=3, küçük pencere ortalama farkı=2

BÖLÜM 5
STOKASTİK GRADYENT MASKESİ

Daha önceki bölümlerde anlatılan kenar tanıma metodları gürültüsüz ortamlar için iyi sonuçlar versede gürültü ortamlar için performansları çok düşüktür. Daha iyi performans için, gürültüyüde hesaba katan sistemlerin kullanılması gerekmektedir. Bu nedenle $f(i,j)$ görüntü fonksiyonunun stokastik modellerinin linear kestirim metodlarının uygulanması gerekmektedir. Bu bölümde Wiener filtresi kullanılarak stokastik gradyent maskeleri metodu incelenecektir.

5.1 2-B Sayisal fonksiyonların Stokastik modelleri.

$u(m,n)$ sıfır ortalamalı kovaryans fonksiyonu $r(k,l)$ olan stasyoner bir rastlantısal bir fonksiyon $u(m,n)$ nin lineer kestirimi $\bar{u}(m,n)$ olsun

$$\bar{u}(m,n) = \sum_{k,l \in \hat{S}} \sum a(k,l)u(m-k,n-l) \qquad (5.1)$$

burada $a(k,l)$ kestirim katsayıları ve $\hat{S}_x$ ise kestirim bölgesidir.

2-B nedensellik bir sorun olmadığı için kestirim alanları aşağıdaki gibi 3 değişik şekilde olabilir.

a. Nedensel kestirim: Sadece o zamana kadarki elemanlarla yapılan kestirimlere denir. Destek bölgesi

$\hat{S}_x = \{l \geq 1, \forall k\} \cup \{l=0, k \geq 1\}$ dir.

b. Yarı-nedensel kestirim: Bir kordinatta nedensel diğerinde ise nedensel değildir. Destek bölgesi

$\hat{S}_x = \{l \geq 1, \forall k\} \cup \{l=0, k=0\}$ dir.

c. Nedensel olmayan kestirim. Her iki kordinattada

nedensel değildir. Kendisi hariç olabilecek bütün değişkenleri kullanabilir. Destek bölgesi

$$\hat{S}_x = \{ \forall (k,l) \neq 0 \}$$

Pratikte sınırlı sayıda komşu eleman kullanılır. Buna kestirim penceresi ,$\hat{W}_x$,denir ve u(m,n) aşağıdaki şekilde bulunabilir.

$$u(m,n) = \sum_{k,l \in W_x} \sum a(k,l)u(m-k,n-l) \qquad (5.2)$$

Tezde yarı-nedensel lineer kestirim ve kestirim penceresi boyutlarıda 5x5 olarak kullanılacaktır.

5.2.1 Wiener Filtresi

Teoride Wiener filtresi sonsuz impuls cevaplıdır. Fakat etkili cevap alanı cismin alanından küçük olduğu için , IIR filtrelere göre daha az işleme gerek duyan ve IIR filtre performansını verebilen optimal Wiener filtreleri yapılabilir.

u(m,n) ve v(m,n) sıfır ortalamalı , rastgele iki dizi olsun. v(m,n) yi kullanarak u(m,n) nin kestirimi $\bar{u}(m,n)$ yi hatanın karesinin beklentisel değerini minimum yaparak elde eden filtrelere Wiener filtresi denir. Wiener filtresi, DFT ve doğrudan konvolusyon yöntemleri ile gerçeklenebilir.

FIR Wiener filtresini konvolüsyon yöntemi ile gerçekleyelim.

$$\bar{u}(m,n) = \sum_{i,j \in W} \sum g(i,j)v(m-k,n-j) \qquad (5.3)$$

burada g(i,j) hatanın karesini minimum yapan Wiener filtresi parametreleridir.

Ortogonality prensibini

$$E\left[(u(m,n)-\bar{u}(m,n))v(m-k,n-l)\right] = 0 \qquad (5.4)$$

ve çapraz korolasyon tanımını kullanarak

$$r_{ab}(m,n;k,l) = E\,[a(m,n)b(k,l)] \qquad (5.5)$$

ortogonality denklemi aşağıdaki gibi yazılabilir.

$$E\,[u(m,n)v(m-k,n-j)\,]-E\,[\,u(\bar{m},n)v(m-k,n-l)] = 0 \qquad (5.6)$$

ve denklem 5.3 deki $\bar{u}(m,n)$ yi ve denklem 5.5 deki çapraz-korolasyon denklemini denklem 5.6 ya yerleştirirsek aşağıdaki sonuçu elde ederiz.

$$r_{uv}(k,l)-\sum_{(i,j)\in W}\sum g(i,j)r_{vv}(k-i,l-j)=0\forall(k,l)\in W \qquad (5.7)$$

$v(m,n)$ nin sıfır ortalamalı , varyansı σ_n^2 olan toplamsal Gaussian beyaz gürültülü, $\eta(m,n)$, bir sistemde modellendiğini düşünelim, yani

$$v(m,n) = \sum_{k,l=-\infty}^{+\infty}\sum h(m-k,h-l)u(k,l)+\eta(m,n) \qquad (5.8)$$

Sistemde bulanıklığın olmadığını kabul edelim $(h(m,n)=1)$ ve $u(m,n)$ ve $\eta(m,n)$ korolasyonsuz oldukları için $r = 0$ olur, ve

$$r_{vv}(k,l) = r_{uu}(k,l)+\sigma_n^2 \qquad (5.9)$$

$$r_{uv}(k,l) = r_{uu}(k,l)$$

olarak yazılabilir. Denklem 5.9 daki r_{vv} ve r_{uv} 'yi deklem 5.7 ye yerleştirirsek

$$r_{uu}(k,l)=\sum_{(k,l)}\sum_{-\infty}^{-\infty} g(i,j)\left[r_{uu}(k-i,l-j)+\sigma_n^2\right] \qquad (5.9)$$

denklem 5.10'u $u(m,n)$'nin varyansı σ 'ya böler ve bu denlemin bir konvolüsyon çarpımı olduğunu görürsek ,denklem 5.10 aşağıdaki gibi yazılabilir.

$$r_o(k,l) = g(i,j)*\left[\frac{\sigma_n^2}{\sigma}\delta(k,l)+r_o(k,l)\right] \qquad (5.11)$$

burada * konvolüsyon işlemini göstermekte ve

$$r_o(k,l) = \frac{r_{uu}(k,l)}{r_{uu}(0,0)} = \frac{r_{uu}(k,l)}{\sigma} \qquad (5.12)$$

olarak kullanılmaktadır.

Matrix formunda bu sistem Toeplitz olduğu için, yukarıdaki denklem aşağıdaki gibi yazılabilir.

$$\left[\frac{\sigma_n^2}{\sigma^2} I + R \right] g = r_{uv} \qquad (5.13)$$

burada R (2M+1)x(2M+1) blok Toeplitz matriks, r_{uv} bilinen $r_o(k,l)$ leri , g ise bilinmeyen filtre katsayılarını taşıyan $(2M+1)^2$x1 boyutlarında matrikslerdir.

Denklem 5.12'de elde edilen g_{ij}'ler stasyoner u(m,n) için kullanılabilir. Fakat kameralarda tarama yöntemi ile elde edilen görüntüler, stasyoner değildir. Eğer , görüntünün ortalaması ve varyansı yavaşça değişiyorsa, Wiener filtre katsayıları aşağıdaki denklemle hesaplanabilir.

$$\bar{g}_{m,n}(i,j) = g_{m,n}(i,j) + \frac{1}{(2M+1)^2} \left[1 - \sum_{(k,l)\in W} \sum g_{m,n}(k,l) \right] \qquad (5.14)$$

Denklem 5.14 deki ikinci toplam (i,j)'ye göre sabittir ve ve (i,j) noktasındaki W bölgesinin içinde kalan stasyoner filtre katsayılarının ortalamasıyla orantılı bir büyüklüktür.

5.2.2 Stokastik Gradient

Şekil 5.1 de gösterilen, geçiş bölgesi 1 piksel olan bir kenar modelini ele alalım. P noktasındaki kenarı tesbit etmek için , o noktadaki yatay gradyenti hesap edelim,

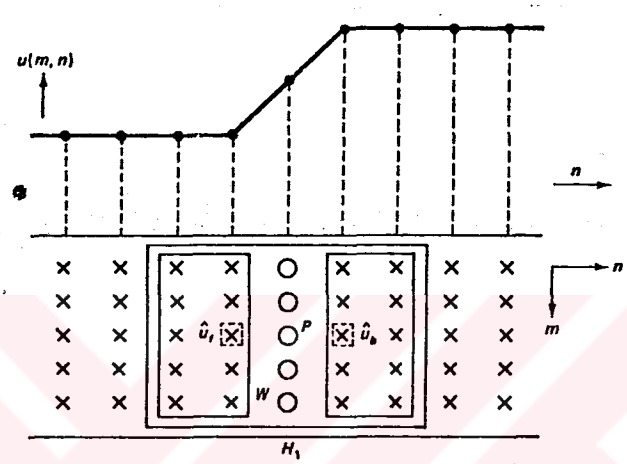
$$\nabla(m,n) = \bar{u}_f(m,n-1) - \bar{u}_b(m,n+1) \qquad (5.15)$$

burada, $\bar{u}_f(m,n)$ ve $\bar{u}_b(m,n)$,sırasıyla u(m,n) 'nin sınırlı bir W bölgesindeki yarı nedensel optimal ileri ve geri

kestirimleridir. Toplamsal beyaz gürültülü gözlemlerden elde edilen v(m,n) nin kestirimini bir önceki bölümde elde edilen FIR Wiener filtresi ile elde edebiliriz. Yani,

$$\bar{u}_f(m,n) = \sum_{(k,l)\in W} \sum a_f(k,l)v(m-k,n-l) \qquad (5.16)$$

burada $W = [(k,l): |k| \le p, 0 \le l \le q]$



Şekil 5.1 1 piksel geçiş bölgeli kenar modeli

İleri kestirim filtre katsayıları $a_f(k,l)$' ler (5.13) ve (5.11) . denklemlerinden elde edilebilir . Burada kullanılan W bölgeleri yarınedensel olarak kullanılacaktır. Aynı şekilde elde edilen geri kestirim katsayıları $a_b(m,n)$, $a_f(m,n)$ ile aynıdır. Denklem (5.15) deki gradyent tanımını kullanarak bulunan yatay maske şekil 5.2 'de gösterilmiştir. Simetri özelliğinden

5x5

0.802	0.836	0	−0.836	−0.802
0.845	0.897	0	−0.897	−0.845
0.870	1.00	[0]	−1.00	−0.870
0.845	0.897	0	−0.897	−0.845
0.802	0.836	0	−0.836	−0.802

SNR=1

0.267	0.364	0	−0.364	−0.267
0.373	0.562	0	−0.562	−0.373
0.463	1.00	[0]	−1.00	−0.463
0.373	0.562	0	−0.562	−0.373
0.267	0.364	0	−0.364	−0.267

SNR=9

$r_0(k,l) = .99^{\sqrt{k^2+l^2}}$

Şekil 5.2 Yatay stokastik gradyent maskesi

dolayı dikey gradyent maskesi yatay maskenin 90° saatin ters yönünde döndürülmüş halidir,yani transpozesidir. Tablo 5.1 de gösterilen maskeler a(0,0)=1 olacak şekilde normalize edilmişlerdir.

5.2.3 Korolasyon Matrisinin Bulunması

Sistemde ayrılabilir kovaryans fonksiyonları kullanılmıştır. Ayrılabilir kovaryans fonksiyonları görüntü işlemede sık sık kullanılırlar ve aşağıdaki formülle ifade edilebilirler,

$$r(m,n) = \sigma^2 \rho_1^m \rho_2^n \qquad (5.17)$$

burada σ^2 görüntünün varyansını ρ_1 ve ρ_2 ise bir adımlık korolasyonları ifade ederler

$$\rho_1 = \frac{r(1,0)}{\sigma^2} \quad \text{ve} \quad \rho_2 = \frac{r(0,1)}{\sigma^2} \qquad (5.18)$$

r(0,1) ve r(1,0) bulunurken sistemim zaman ortalamaları kullanılmıştır ve aşağıdaki şekilde ifade edilirler,

$$\mu = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} u(m,n) \qquad (5.19.a)$$

$$r(m,n) = \frac{1}{MN} \sum_{m'=1}^{M-m} \sum_{n'=1}^{N-n} [u(m',n')-\mu][u(m'+m,n'+n)-\mu] \qquad (5.19.b)$$

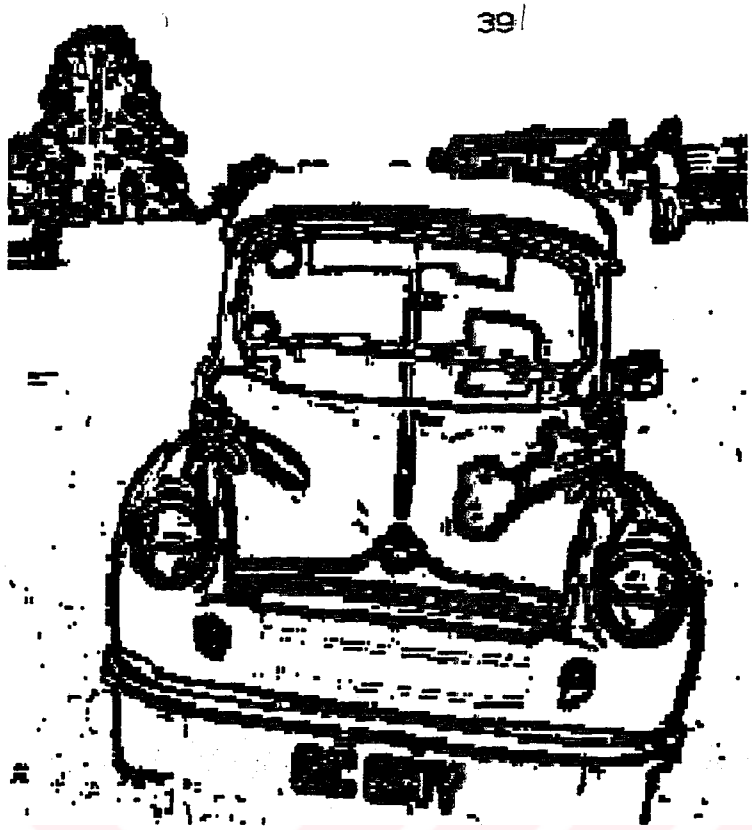
ρ_1 ve ρ_2, birçok görüntü için .95 civarındadır. Kullanılan test görüntüleri için .9 ile 1.0 arasında bulunmuştur.Bölüm 7 de gürültülü ortamlarda ise korolasyon katsayıları dahada düşmektedir.

5.3 Sonuç

Gradient maskeleri ile elde edilen sonuçlar diğer metodlara göre belirgin bir üstünlük sağlamaktadır. Şekil 5.3 de test resimlerinin elde edilen kenar görüntüleri verilmiştir. Kullanılan test resimleri gürültüsüz kabul edilmiş ve kenar görüntüleri σ_n=0 için elde edilmiştir.

Maske paremetreleri, sistemin gürültü ve bulanıklık özelliklerine göre hesaplandığı için sistem çok iyi performans göstermektedir. Çok yüksek gürültü oranlarında kenarları doğru olarak tesbit edebilmektedir.

Sistemin daha iyi kenar görüntüleri vermesinin diğer bir sebebi de 5x5 boyutlarında maskeler kullanmamızdır. 3. bölümdeki yöntemlerde de 5x5 veya daha yuksek mertebelerde maskeler kullanılsa daha iyi sonuçlar elde edilebilirdi.



Şekil 5.3.a Stokastik gradyent metodu, eşik=4

Şekil 5.3.b Stokastik gradyent metodu, eşik=4

BÖLÜM 6
FIR KENAR TANIMA FİLTRESİ

Gürültülü ortamlar için daha iyi sonuçlar verebilen diğer bir yöntemde filtreleme metodları kullanmaktır[6]. Bu bölümde, belirli çözünürlük aralığında kenar bölgelerinde enerjiyi maksimum yapan bir filtre incelenecektir. Filtrenin çözünürlük ve band genişliği çarpımı istenilen şekilde seçilebilmektedir. Filtre birinci dereceden prolate spheroidal dalga fonksiyonlari ile ifade edilmiş ve fourier metodu ile gerçeklenmiştir.[7].

6.1 Filtrenin Tanımlanması

Kenar tanıma 1 boyutta tanımlanmış ve görüntünün herbir satır ve sütünu için ayrı ayrı uygulanarak bütün görüntünün kenar görüntüsü elde edilmektedir. Filtre kenar bölgelerindeki enerjiyi maksimum yapan frekans domeni filtresidir.

$f(x)$ giriş görüntüsü, $g(x)$ çıkış görüntüsü ve $h(x)$ filtrenin impuls cevabı olsun. $F(w)$, $G(w)$ ve $H(w)$ bunların sırasıyla fourier dönüşümlerini temsil etmektedir. Verilen I ve Ω değerleri için frekans domeni filtresi , aşağıdaki γ parametresini maksimum yapmaya çalışmaktadır.

$$\gamma = \frac{\int_{-I/2}^{I/2} |g(x)^2| \, dx}{\int_{-\infty}^{\infty} |g(x)^2| \, dx} \qquad (6.1)$$

ve bunu yaparken aşağıdaki kısıtlamaları yerine

getirmektedir;

$$F(w) = 1/jw + \delta(w)/2 \qquad (6.2.a)$$

$$G(w) = H(w)F(w) \qquad (6.2.b)$$

$$H(w) = 0 \ , \ w>0 \text{ için} \qquad (6.2.c)$$

$$H(0) = 0 \qquad (6.2.d)$$

$$H(-w) = H(w) \qquad (6.2.e)$$

buradaki maksimum yapmak istediğimiz γ değeri çıkış görüntüsünün toplam enerjisinin I aralığındaki bölümüdür. Bu nedenle I filtrenin çözünürlük katsayısıdır. 1.kısıtlama sistemi giriş basamak fonksiyonunun fourier dönüşümüdür. 2.si sistemi lineer ve değişmeyen filtreyi ifade eder. 3.sünde filtrenin band genişliği Ω ile sınırlanmıştır. 4.sünde filtre çıkışının sabit veya yavaş değişen fonksiyonlar için sıfır olmasını gerektirir ve sonuncusunda filtrenin simetrik olduğunu gösterir. 4.kısıtlamadan dolayı 1.kısıtlamadaki basamak fonksiyonu dönüşümündeki 2.terim fonksiyonu önemini yitirir.

6.2 Filtre Transfer Fonksiyonunun Bulunması

Problemin çözümü prolate küresel dalga fonksiyonlarının kullanılması ile çözülmektedir. prolate küresel dalga fonksiyonları $(-\infty,\infty)$ aralığında olduğu kadar, verilen sınırlı bir aralıktada ortoganal olma özelliğine sahip band sınırlı fonksiyonlar setidir. Bu fonksiyonların belirli özellikleri aşağıdadır.

Verilen $I>0$ ve $\Omega>0$ değerleri için aşağıdaki 3 özelliğe sahip sayısız $\psi_0(x)$, $\psi_1(x)$, $\psi_2(x)$, $\psi_3(x)$ ... reel fonksiyonları ve $\lambda_0>\lambda_1>\lambda_2>\lambda_3...$ pozitif reel sayıları bulunabilir.

1) $\psi_i(x)$ sınırlı ve band sınırlı fonksiyonlar uzayında kapalı bir settirler.

2) $\psi_i(x)$ler $(-\infty,\infty)$reel aralığında ortonormal ve $-I/2$

$<x\leq I/2$ aralığında ise ortogonaldirler. Ayrıca $L^2_{I/2}$de kapalıdırlar.

$$\int_{-\infty}^{\infty} \psi_i(x)\psi_j(x)dx = \begin{cases} 0 & , i\neq j \\ 1 & , i=j \end{cases} \qquad i,j=0,1,2.. \qquad (6.3.a)$$

$$\int_{-I/2}^{I/2} \psi_i(x)\psi_j(x)dx = \begin{cases} 0 & , i\neq j \\ \lambda_i & , i=j \end{cases} \qquad i,j=0,1,2.. \qquad (6.3.b)$$

3) x in reel veya kompleks değerleri için ,

$$\lambda_i\psi_i(x) = \int_{-I/2}^{I/2} \frac{\sin\Omega(x-s)}{\pi(x-s)}\psi_i(x)ds \qquad ,i=0,1,2.. \qquad (6.4)$$

$\lambda_i$ve $\psi_i(x)$ bir c sabitinin fonksiyonudur, ve $2c=\Omega I$ olarak tanımlanır.

Prolate küresel dalga fonksiyonlarını kullanarak filtrenin çıkışı ,g(x), aşağıdaki gibi yazılabilir.

$$g(x) = \sum_{n(tek)} a\psi(c_n,x) \qquad (6.5)$$

n tek sayi olarak sınırlandırılmıştır. Çünkü F(w) tektir ve bu yüzden çift fonksiyon H(w) ve F(w) çarpımının ters fourier dönüşümü olan g(x) tek fonksiyon olur.

$$\gamma = \frac{\sum_{n(tek)} |a_n|^2\lambda_n}{\sum_{n(tek)} |a_n|^2} \qquad (6.6)$$

λ_n ler pozitif ve $\lambda_1>\lambda_2>\lambda_3>...$olduğu için γ, $0<\gamma\leq\lambda_1$ ile sınırlıdır ve denklem 6.6 dan görülebileceği gibi γ'nin maksimum değeri $n>0$ için $a_n=0$ olduğu zaman olmaktadır . Yani maksimum çıkış

$$g(x) = a\,\psi(c,x) \qquad (6.7)$$

$\psi_i(c,x)$in fourier dönüşümü aşağıdaki denklemle bulunabilir,

$$jK\psi_i(c,x) = \frac{1}{2\pi}\int_{-\Omega}^{\Omega} \exp(jwx)\psi_i(c,wI/2\Omega)dw \qquad (6.8)$$

Denklem 6.7 ve 6.8 birleştirilirse , optimum filtre

transfer fonksiyonu aşağıdaki gibi bulunabilir.

$$H_{opt}(w) = \frac{G(w)}{F(w)} = \begin{cases} K_1 \dfrac{\psi_1(c, wI/2\Omega)}{jF(w)} & ,|w|<\Omega \\ 0, & ,\text{başkayerlerde} \end{cases} \qquad (6.9)$$

F(w) yı denklem 6.9'a yerleştirirsek, $H_{opt}(w)$

$$H_{opt}(w) = \begin{cases} K_1 w \psi_1(c, wI/2\Omega) & ,|w|<\Omega \\ 0, & ,\text{başkayerlerde} \end{cases} \qquad (6.10)$$

olarak bulunur. Burada K_1 reel bir sabittir.

Denklem 6.10 sade gibi görünse bile $\psi_i(c,x)$ 'in hesaplanması oldukça zordur. Oldukça fazla işlem sayısı ve hafıza gerektirmektedir. Bunun yerine $\psi_i(c,x)$ in oldukça iyi bir yaklaşımı olan aşağıdaki ifade kullanılacaktır.[ ]

$$\psi_n(c,x) = \left[\frac{c}{2}\right]^{1/4} 2^{-n/2}(n!)^{-1/2} H_n(c^{1/2}x)\exp(-cx^2/2) \qquad (6.11)$$

burada $H_n(x)$ n. dereceden Hermit polinomudur.

$H_1(x)= 2x$ için , optimum filtrenin transfer fonksiyonu aşağıdaki gibi olur.

$$H_{opt}(w) = \begin{cases} \dfrac{(K_1 w)\exp[-(cI^2w^2)/4\Omega^2]}{jF(w)} & ,|w|<\Omega \\ 0, & ,\text{başkayelerde} \end{cases} \qquad (6.12)$$

F(w) basamak fonksiyonu için , $H_{opt}(w)$

$$H_{opt}(w) = \begin{cases} (K_1 w)\exp[-(cI^2w^2)/4\Omega^2] & ,|w|<\Omega \\ 0, & ,\text{başkayerlerde} \end{cases} \qquad (6.13)$$

olarak en son şeklini alır.

6.3 Filtrenin Gerçeklenmesi

FIR filtreleri gerçekleştirmede en kolay metod

konvolüsyon toplamı veya FFT yöntemidir. Filtrenin çıkışı aşağıdaki DFT formulüyle bulunabilir.

$$G(w) = H(w)F(w) \qquad (6.14)$$

buradan g(x) , ters DFT dönüşümüyle elde edilir.

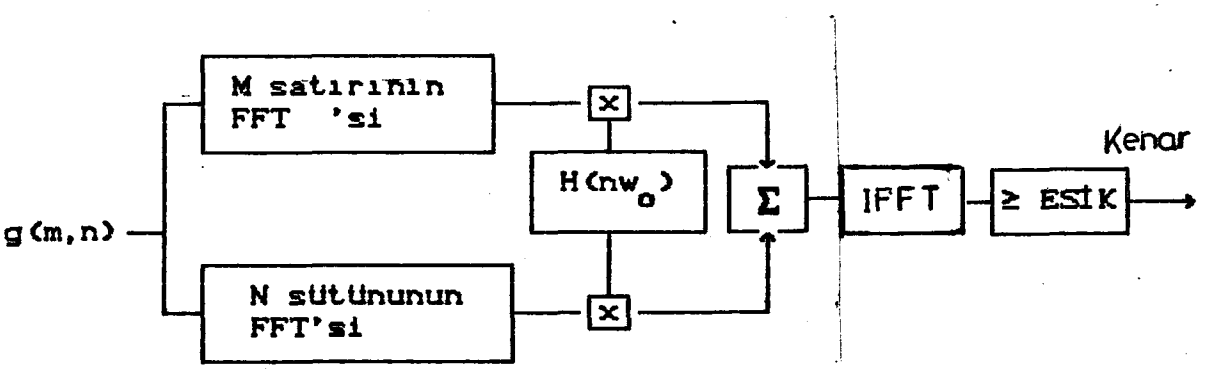
$$g(x) = F^{-1}(H(w)F(w)) \qquad (6.15)$$

burada F(w) giris görüntüsünün fourier dönüşümü ve H(w)

$$H_{opt}(nw_o)=\begin{cases}\dfrac{(K_1 nw_o)^2 \exp[-(c^3 n^2 w_o^3)/2\Omega^4]}{jF(nw_o)} & ,|w|<\Omega \\ 0, & ,\text{başkayerlerde}\end{cases} \qquad (6.16)$$

burada $w_o = 2\pi/(2N+1)$ ve $c=\Omega I/2$ olarak kullanılmıştır.

Tezde herbir satır ve sütün ayrı ayrı ele alınmış. Bir tanesinin fourier dönüşümü alınmış , $H(nw_o)$ ile çarpılmış ve ters fourier dönüşümü alınarak g(n) bulunmuştur. Bu işlem herbir satır ve sütün için tekrarlanmıştır. Böylece herbir görüntü noktası için biri yatay diğeri dikey doğrultudan olmak üzere iki enerji katsayısı elde edilmiştir. Bu iki büyüklüğün mutlak değerleri toplanarak, belli bir eşik değeri ile karşılaştırılması ile kenar görüntüsü elde edilmiştir.



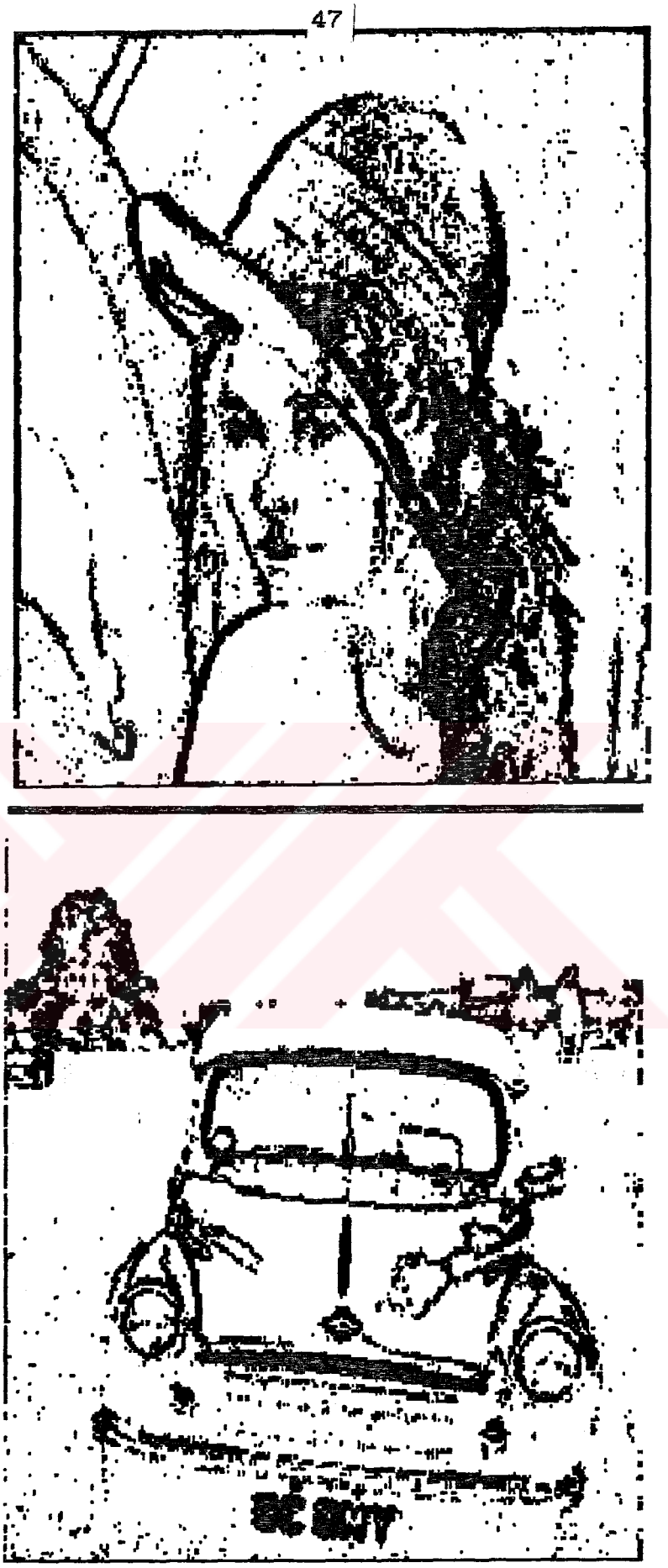
Şekil 6.1 FFT metodu ile gerçekleştirmenin blok şeması.

6.4 Sonuç

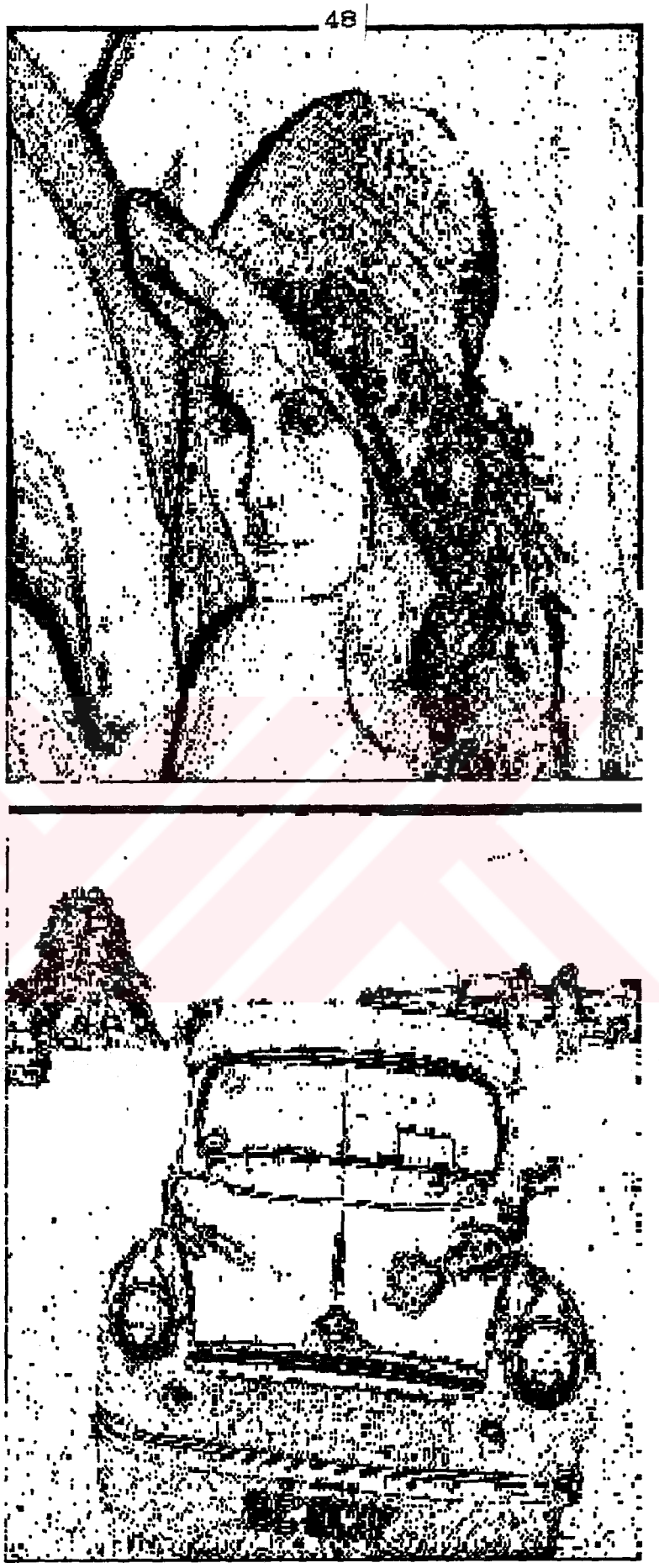
Algoritma 2 test görüntüsüne uygulanmıştır. Algoritma c katsayısına bağlıdır. C katsayısı sistemin band genişliği-çözünürlük katsayısıdır. Küçük c katsayısı için I da küçük olacağı için kenarlar daha hassas tanınabilecektir. Fakat gürültüye daha hassas duruma gelecektir. Gürültülü Lena ve araba resimlerinde ise kenar görüntüsü ancak daha büyük c kaysayısı ile elde edilebilmektedir ve kenar hassasiyeti ise daha kötüdür. Dolayısı ile c katsayısı kenarların hassas tesbiti ile kenarların tanınabilmesi arasında bir seçimi gerektirmektedir. Şekil 6.2 c katsayısının 3 değişik değeri için elde edilen kenar görüntülerinde bu durum görülmektedir. c=1.5 için net bir görüntü elde edilmiştir. Fakat c=3.0 ve c=5.0 için kenar bölgeleri genişlemiştir.

Gürültüsüz resimler için küçük c katsayıları yeterli olmaktadır. Fakat görüntüdeki impuls gürültü ve bulanıklaşma oranı arttıkça c katsayısı büyütülerek kenarlar tesbit edilmektedir. C katsayısı büyütüldüçe gürültüsüz resimler için daha kötü bir kenar görüntüsü oluşturmakta , fakat gürültü oranı arttıkça sistemin performansı diğer geleneksel metodları geçmektedir.

Şekil 6.2 FIR filtre metodu, c=1.5



Şekil 6.3 FIR filtre metodu, c=3.0



Şekil 6.4 FIR filtre metodu, c=5.0

BÖLÜM 7
MORFOLOJİK KENAR TANIMA METODLARI

Morfoloji yapı ve formlarla uğraşmak demektir. Morfoloji, önceden belirlenmiş bir şekli olan ve yapı elemanı olarak adlandırılan iki değerli görüntü ile çalışır. Bu yapı elemanının orijinal görüntüyle olan ilişkisi incelenir. Morfolojik operasyonlar , kenar tanıma , segmentlere ayırma ve görüntünün iyileştirilmesi gibi birçok alanda başarılı olarak kullanılabilirler. [8] Fakat bugüne kadar üzerinde çok az çalışma yapılmıştır. [9]. Bu bölümde ilk önce temel morfolojik kenar operatörleri incelenecek ve bunları kullanarak geliştirilmiş bir operator gerçekleneçektir.

7.1 Temel Morfolojik Operasyonları

Morfolojik operasyonlar iki görüntü ile çalışırlar. Birisi orijinal görüntü verileri , diğeri ise yapı elemanıdır. Herbir yapı elemanının şekli operasyonların bir paremetresi gibi düşünülebilir. İki temel morfolojik operasyonu inceleyeceğiz, erozyon ve dilasyon.

Binary sistemde, A orijinal görüntünün 1 olan piksellerini, B de yapı elemanının 1 olan piksellerini göstersin. A'nın B tarafından dilasyonu ,A⊕B, aşağıdaki gibi ifade edilir,

$$A \oplus B = \bigcup_{a \in A} \{ b+a \setminus b \in B \} \qquad (7.1)$$

ve A'nin b tarafından erozyon'u, A⊖B, ise aşağıdaki gibi tanımlanır.

$$A \ominus B = \{p \setminus B+p \subseteq A\} \qquad (7.2)$$

Gri skalası sisteminde,gri seviyeli görüntü, f 'in gri seviyeli yapı elemanı, b tarafından dilasyonu, d ile gösterilir ve aşağıdaki gibi tanımlanabilir.

$$d(r,c) = \max_{i,j} (f(r-i,c-j)+b(i,j)) \qquad (7.3)$$

burada , maksimum operasyonu , (r-i,c-j) toplamı f domeninde olacak şekilde b domenindeki bütün (i,j) noktalarında tanımlanır.

Gri seviyeli f görüntüsünün , gri seviyeli b yapı elemanı tarafından erozyonu e ile tanımlanır ve aşağıdaki gibi tanımlanır.

$$e(r,c) = \min_{i,j} (f(r+i,c+j)-b(i,j)) \qquad (7.4)$$

burada minimum operasyonu b domenindeki bütün (i,j) noktalarında alınır.

7.2 Morfolojik Kenar Tanıma Operatörü

Morfolojik sistemde kenarları tanımanın en kolay yolu , görüntünün çubuk şeklinde küçük bir yapı elemanı ile erozyonunu olmaktadır. Fark görüntüsü kenar kuvveti görüntüsünü verir. Uygun eşik değeri ile kenar görüntüsünü elde ederiz.

Merkez noktasını (0,0) olarak alalım. Çubuk şeklindeki yapı elemanını, tepesi düz ve disk şeklinde domeni olacak şekilde aşağıdaki gibi tanımlanabilir.

$$D_{\text{çubuk1}} = \{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(-1,1),(-1,1),(1,1)\} \qquad (7.5)$$

$b: D_{\text{çubuk1}} \longrightarrow \{0,\ldots 255\}$ rodl yapı elemanı olsun. Çubuk tepede düz olduğu için bütün $b(r,c)$, $(r,c)\in D_{\text{çubuk1}}$ lerin gri seviyeleri sıfır olur. $b(r,c)$ ler sıfır olduğu için $f(r,c)$ nin erozyonu ,

$$e(r,c)= \min_{(i,j)\in D_{\text{çubuk1}}} [f(r+i,c+j)-b(r,c)] = \min_{(i,j)\in D_{\text{çubuk1}}} [f(r+i,c+j)] \qquad (7.6)$$

Erozyon kenar tanıma operatörü, G_e, aşağıdaki gibi tanımlanır,

$$G_e(r,c) = f(r,c)-e(r,c) = f(r,c)-\min_{(i,j)\in D_{çubuk}} f(r+i,c+j)$$

$$= \max_{(i,j)\in D_{çubuk}} [\, f(r,c)-f(r+i,c+j)]$$ (7.7)

Elde ettiğimiz, kenar kuvveti operatörü, (r,c) pikseli etrafındaki 8 komşu nokta ile arasındaki farkların maksimumunu alan doğrusal olmayan bir operatördür. Lineer Laplas operatörüne (Şekil 3.9.b) çok benzeyen bir yapıdadır. Gerçek basamak kenara (u(n) = 1, n≥0 ve u(n)=0 , n<0) karşı G_e ve $\nabla^2 f$ cevapları aşağıdaki gibidir.

Yönler		0°	45°	90°	135°
$G_{e(8\text{-komsulu})}$	=	1	1	1	1
$\nabla^2 f$	=	3	3	3	3

Erozyon morfolojik kenar operatörü Laplas benzeri doğrusal olmayan bir operatordur. Aynı zamanda Laplas kadar olmasada gürültüye karşı hassas bir operatördür. Operatörün destek bölesini artırdıkça, gürültü hassasiyeti artmaktadir. Şekil 7.1.a deki 5x5 boyutlarındaki görüntüye erosyon kenar operatörünü uyguluyalım. Merkez noktasındaki impals gürültüye uygulanırsa şekil 7.1.b deki görüntü elde edilir. Eğer yapı elemanının domenini genişletirsek , merkezdeki impuls gürültüden dolayı hatalı kenar noktaları dahada artar. Erosyon kenar operatörü kenar bölgelerinde gri seviyesi yüksek olan tarafı kenar noktası olarak belirler.

E E E E E	0 0 0 0 0
E E E E E	0 E E E 0
E E 0 E E	0 E 0 E 0
E E E E E	0 E E E 0
E E E E E	0 0 0 0 0
(a)	(b)

Şekil 7.1 Bir piksel gürültülü görüntü ve kenar görüntüsü

Kenar geçiş bölgelerinin her iki tarafınıda kenar olarak tanımlamak için erozyon operatörü , dilasyon operatörü ile birlikte kullanılabilir. Dilasyon operatörü, görüntünün dilasyon'u ile orijinal görüntünün farkınını alır. Eğer erozyon'da kullandığımız çubuk yapı elemanını kullanırsak, gri seviyeli görüntü $f(r,c)$ nin dilasyon'u

$$d(r,c) = \max_{(i,j)\in D_{çubuk}} [\, f(r-i,c-j)] \qquad (7.8)$$

ve kenar kuvveti görüntüsü

$$G_d(r,c) = d(r,c) - f(r,c) = \max_{(i,j)\in N(r,c)} [\, f(i,j) - f(r,c)] \qquad (7.9)$$

ile ifade edilir. $N(r,c)$ yapı elemanının destek bölgesini göstermektedir.

Kenarın her iki tarafınıda bulan bir oparatör, $G_e(r,c)$ ve $G_d(r,c)$ nin piksel piksel minimum, maksimum veya toplamından elde edilebilir. İlk olarak maksimum operasyonunu ele alalım.

$$E(r,c) = \max (G_e(r,c) , G_d(r,c)) = \max_{(i,j)\in N(r,c)} (f(r,c) - f(r,c)) \qquad (7.10)$$

olarak tanımlansın. Bu operatörün performansını aşağıdaki örnekte inceleyelim. Giriş görüntüleri olarak şekil 7.2 deki görüntüleri alalım.

```
E E E E E      E E E O O      E E E E E
E E E E E      E E E O O      E E E E O
E E E E E      E E E O O      E E E O O
O O O O O      E E E O O      E E O O O
O O O O O      E E E O O      E O O O O
   (a)            (b)            (c)

E E E E E      O O E/2 E E    O O O O O
O E E E E      O O E/2 E E    O O O O O
O O E E E      O O E/2 E E    O O N O O
O O O E E      O O E/2 E E    O O O O O
O O O O E      O O E/2 E E    O O O O O
   (d)            (e)            (f)
```

Şekil7.2 Değişik yönlerde kenar görüntüleri

8 komşu kenarlı yapı elemanı ile şekil 7.3 deki kenar kuvveti görüntüleri elde edilir.

```
O O O O O      O O E E O      O O O O E
O O O O O      O O E E O      O O O E E
E E E E E      O O E E O      O O E E O
E E E E E      O O E E O      O E E O O
O O O O O      O O E E O      E E O O O
   (a)            (b)            (c)

E O O O O      O E/2 E/2 E/2 O      O O O O O
E E O O O      O E/2 E/2 E/2 O      O O N O O
E E O O O      O E/2 E/2 E/2 O      O N N N O
E E O O O      O E/2 E/2 E/2 O      O O N O O
E E O O O      O E/2 E/2 E/2 O      O O O O O
   (d)            (e)            (f)
```

Şekil 7.3 Maksimum operatörü kenar görüntüsü

Sonuçtan görüldüğü gibi kenarların her iki tarafıda tanınmıştır. Fakat rampa kenarlar eşit kuvvette olarak daha da genişlemiştir. Ayrıca gürültüye karşı hassasdır. Yani rampa kenar ve gürültüler için kötü bir operatördür.

Şimdi de $G_e(r,c)$ ve $G_d(r,c)$ nin toplamına bakalım.

$$E(r,c) = G_e(r,c)+G_d(r,c) \qquad (7.11)$$

Bu operatörün performansını görmek için Şekil 7.2 deki görüntülere uygulayalım. Sonuçlar şekil 7.4 de verilmiştir.

```
O O O O O      O O E E O      O O O O E
O O O O O      O O E E O      O O O E E
E E E E E      O O E E O      O O E E O
E E E E E      O O E E O      O E E O O
O O O O O      O O E E O      E E O O O
   (a)            (b)            (c)

E O O O O      O E/2 E E/2 O      O O O O O
E E O O O      O E/2 E E/2 O      O O N O O
O E E O O      O E/2 E E/2 O      O N N N O
O O E E O      O E/2 E E/2 O      O O N O O
O O O E E      O E/2 E E/2 O      O O O O O
   (d)            (e)            (f)
```

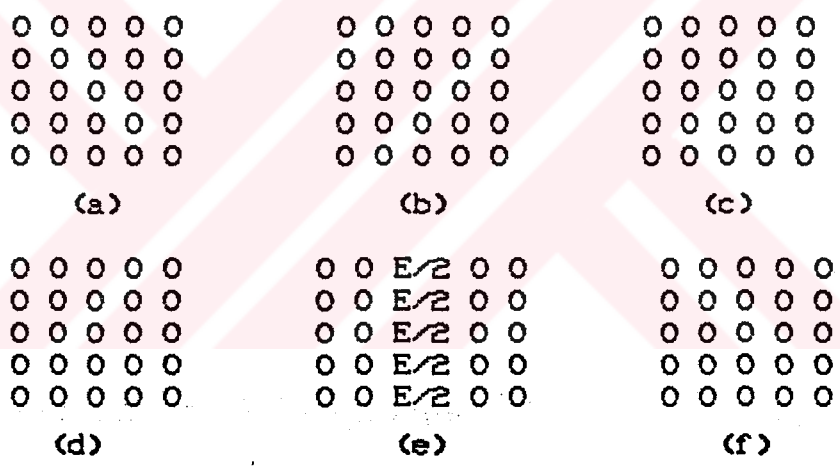
Şekil 7.4 Toplam operatörü kenar görüntüleri

Toplama operatörü gürültü ve basamak kenara maksimum ile aynı cevap vermiştir. Fakat rampa kenarlar eşit ağırlıklı olmadığı için uygun eşik değeri ile tanınabilirler.

Son olarakta minimum operatörünü inceleyelim .

$$E(r,c) = \min \{G_e(r,c), G_d(r,c)\} \qquad (7.12)$$

Sonuçlar şekil 7.5 de gösterilmiştir. Sonuçlar oldukça ilginçtir. Operatör gürültüye duyarsız olmuş ve rampa kenarları daha iyi tanımaktadır. Fakat ideal basamak kenarı tanıyamamaktadir. Bu özelliği aşağıdaki bölümde geliştirilmiş morfolojik operatör için kullanılacaktır.



Şekil 7.5 Minimum operatörü kenar görüntüsü

7.3 Minimum ve Bulanık Operatörü

Mimimum operatörü rampa kenarları iyi tanıdığı ve gürültülü görüntüler için daha iyi olduğu için , önce görüntüyü bulanıklaştırıp , sonra minimum operatörünü uygularsak iyi bir kenar görüntüsü elde edebiliriz. Bu morfolojik operatör minimum ve bulanık operatörü olarak adlandırılabilir ve aşağıdaki gibi tanımlayabiliriz.

$$I_{kenarkuvveti} = \min \{I_1 - erosion(I_1), dilation(I_1) - I_1\} \qquad (7.13)$$

burada I_1 bulanıklaştırılmış orijinal görüntüdür. Bu operatörü daha iyi anlamak için aşağıdaki basamak kenara uygulayalım

Orijinal görüntü	0	0	0	0	E	E	E	E
I_1	0	0	0	E/3	2E/3	E	E	E
Erozyon (I_1)	0	0	0	0	E/3	2E/3	E	E
Dilasyon (I_1)	0	0	E/3	2E/3	E	E	E	E
I_1 - erozyon(I_1)	0	0	0	E/3	E/3	E/3	0	0
Dilasyon(I_1) - I_1	0	0	E/3	E/3	E/3	0	0	0
Kenar kuvveti	0	0	0	E/3	E/3	0	0	0

Görüldüğü gibi basamak kenarı oldukça iyi tanımıştır. Şimdi de aşağıdaki gürültü için uygulayalım.

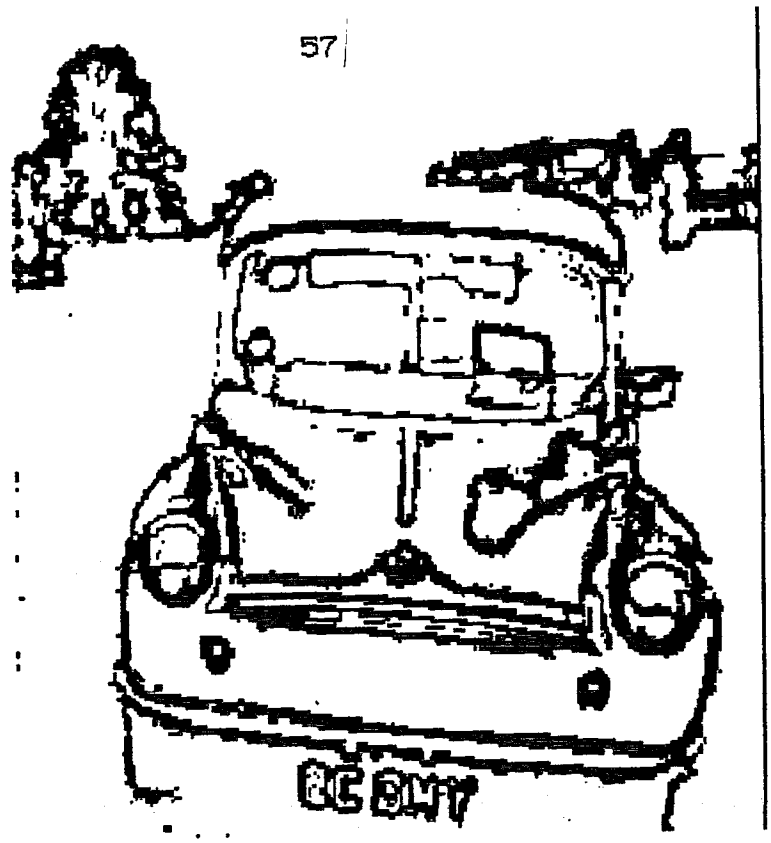
Orijinal görüntü	0	0	0	0	N	0	0	0
I_1	0	0	0	N/3	N/3	N/3	0	0
Erozyon (I_1)	0	0	0	0	N/3	0	0	0
Dilasyon (I_1)	0	0	N/3	N/3	N/3	N/3	N/3	0
I_1 - erozyon(I_1)	0	0	0	N/3	0	N/3	0	0
Dilasyon(I_1) - I_1	0	0	N/3	0	0	0	N/3	0
Kenar kuvveti	0	0	0	0	0	0	0	0

Görüldüğü gibi impuls gürültü tanınmamıştır.

7.4 Sonuç

Minimum ve bulanık metodu ile gerçekleştirilen kenar görüntüleri şekil 7.6 da verilmiştir. Gürültüsüz görüntüler için sonuçlar tatmin edicidir. Fakat gürültülü görüntüler için sonuçlar yetersizdir. Görüntülerde dikkat edilen nokta , bulanıklaştırmadan dolayı kenarlarda gerçek yerlerinden kayma görülmektedir. Ayrıca rampa kenarlar için uygun eşik değer , gürültülü ortamlar için zor olmakta , bu da operatörün performansını düşürmektedir.

Gerçeklemede 3x3 boyutlarında bulanıklaştırma operasyonu ve 5x5 boyutlarında da erozyon ve dilasyon operasyonları uygulanmıştır.



Şekil 7.6.a Minimum bulanık operatorü, eşik=2

Şekil 7.6.b Minimum bulanık opeatörü, eşik=2

BÖLÜM 8
KENAR TANIMA METODLARININ KARŞILAŞTIRILMASI

Kenar tanıma algoritmalarının performansları birçok değişik şekilde karşılaştırılabilir. Bu karşılaştırma matematiksel olarak yapılabilir. Fakat , bu matematiksel karşılaştırma bir noktaya kadardır. Seçilen algoritma paremetrelerine bağlı olarak aynı kenar algoritmasının performansı iyi veya kötü çıkabilir.

İlk olarak, kenar tanıma algoritmaları gözle karşılaştırılabilir. Algoritmaların çıkışları görsel olduğu için değişik tekniklerin performansları gözle bakarak karşılaştırılabilir. Aslında bu iyi bir karşılaştırma yöntemidir.

Matemetiksel olarak, gürültülü ortamlar için bir karşılaştırma yöntemi şu şekilde yapılabilir. n_o gürültüsüz görüntü için kenar olarak tanımlanan kenar noktaları ve n_1 gürültü katıldıktan sonraki kenar olarak tanımlanan yeni kenar noktaları veya n_o da tanımlanıpta şimdi tanımlanmayan kenar noktaları olsun. Sistemin performansı aşağıdaki formülle bulunabilir,

$$P_1 = \frac{n_o}{n_1} \qquad (8.1)$$

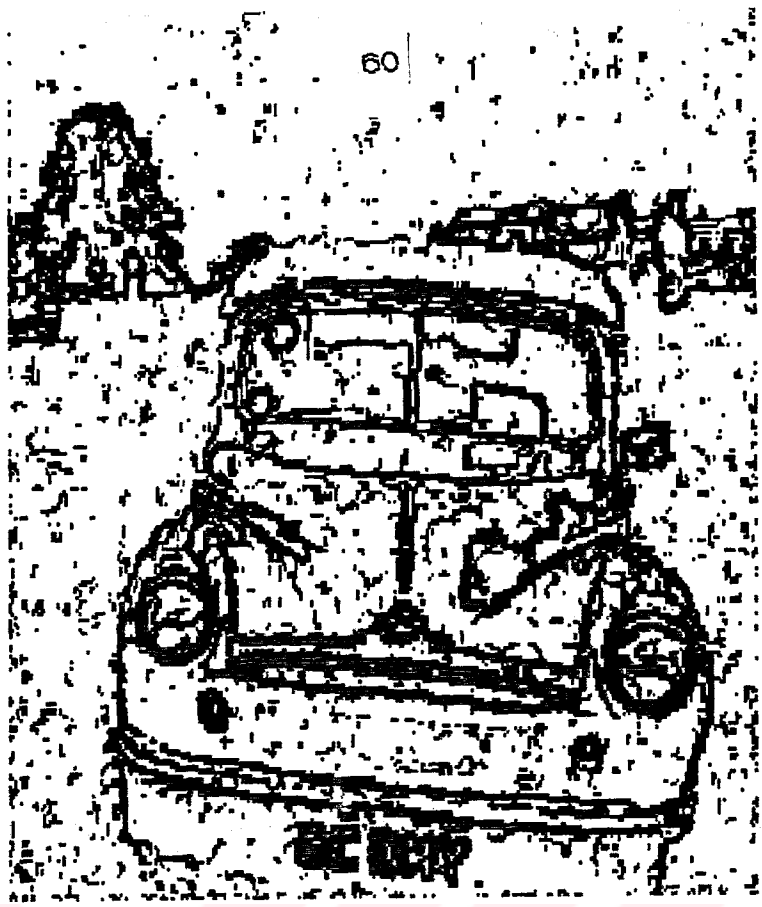
Diğer bir karşılaştırma yöntemide aşağıdaki gibi

$$P = \frac{1}{\max\,(N_1, N_D)} + \sum_{i=1}^{N_D} \frac{1}{1 + \alpha d_i^2} \qquad (8.2)$$

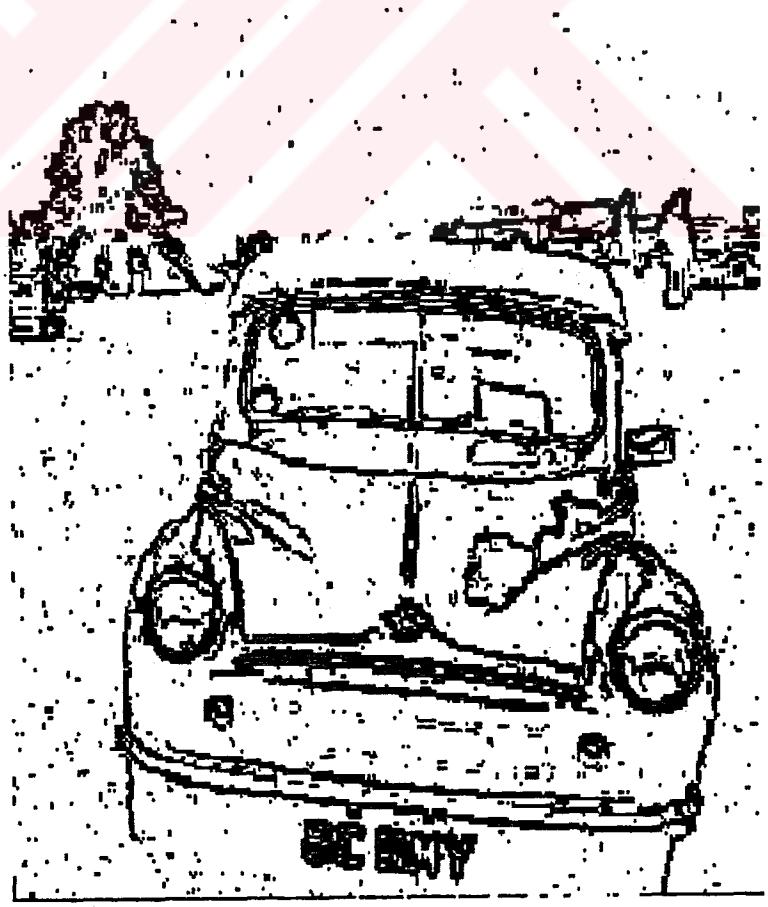
burada d_i gerçek kenar noktası ile buna en yakın kenar olarak tanımlanan nokta arasındaki uzaklıktır. a, ayar

sabitidir ve N_1 , gerçek kenar noktaları N_D ise, kenar olarak tanımlanan kenar noktalarının sayısını verir.

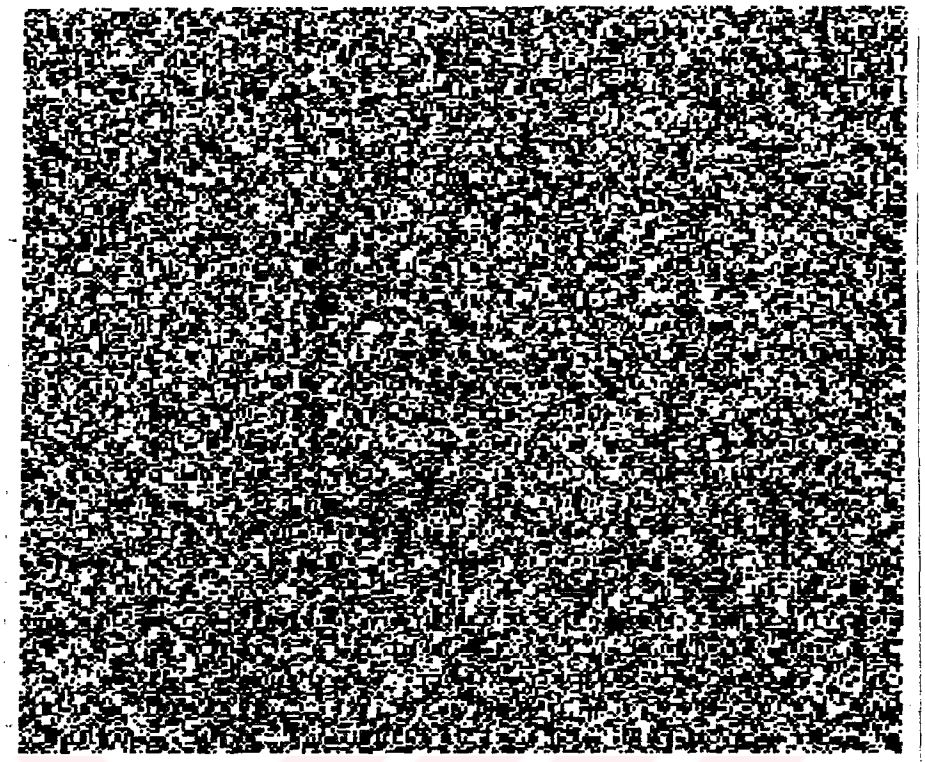
Gürültülü ortamlarda kenar tanıma metodlarının performansları karşılaştırılacağı için denklem 8.1 , ölçü olarak kabul edilecektir. Sinyal-gürültü oranı 10,5,3,2 ve 1 olacak şekilde orijinal görüntüye toplamsal beyaz gürültü eklenecektir. Bu testte sistemin kenarları ne kadar doğru olarak tanıdığı değil, gürültü oranı arttıkça kenarları tanımadaki kararlılığı ölçülmektedir. Şekil 8.1 den şekil 8.8'e kadar sinyal-gürültü oranı 1 olan gürültülü resimlerin kenar görüntüleri gösterilmektedir. Görüntülerden görüldüğü stokastik gradyent metodu en iyi performansı vermektedir. Şekil 8.9 dan şekil 8.14 'e kadar olan grafiklerde değişik sinyal-gürültü oranlarında sistemin kararlılığı gösterilmektedir.



Şekil 8.1 Sobel metodu kenar görüntüsü, SNR=1

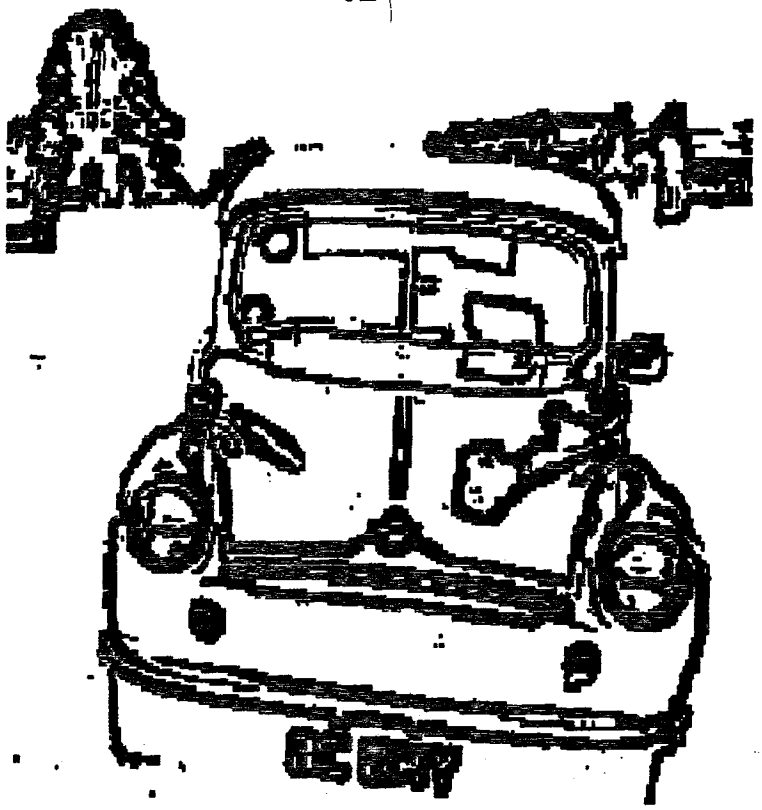


Şekil 8.2 Kompass metodu kenar görüntüsü, SNR=1



Şekil 8.3 Laplasyan metodu kenar görüntüsü, SNR=1

Şekil 8.4 LoG metodu kenar görüntüsü , SNR=1

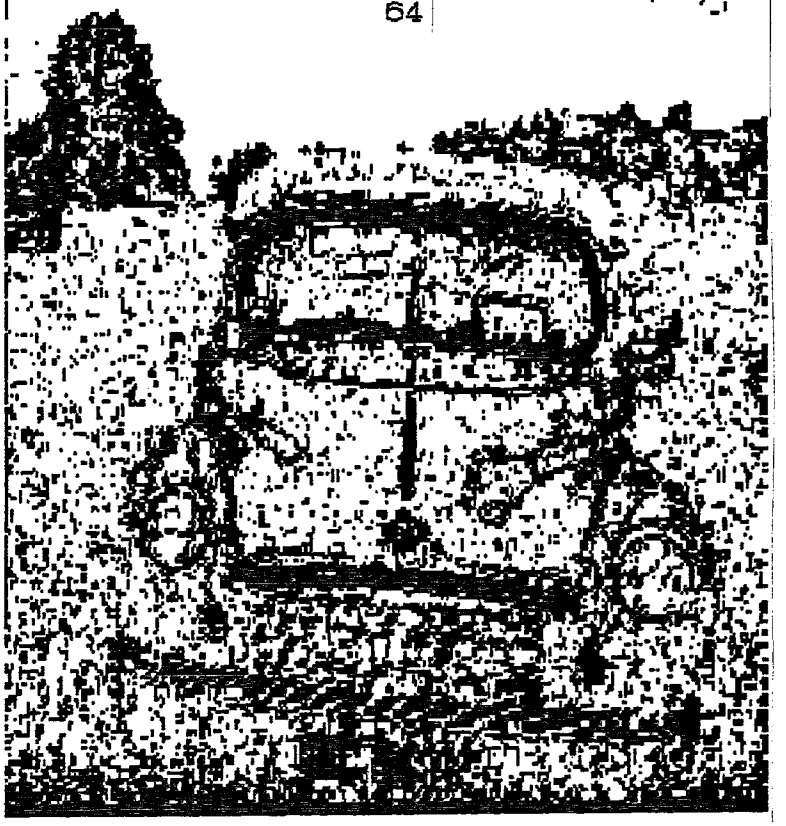


Şekil 8.6.a Stokastik gradyent metodu , SNR=1

Şekil 8.6.b Stokastik gradyent metodu, SNR=1

Şekil 8.5.a Yöresel İstatistik Metodu , SNR=1

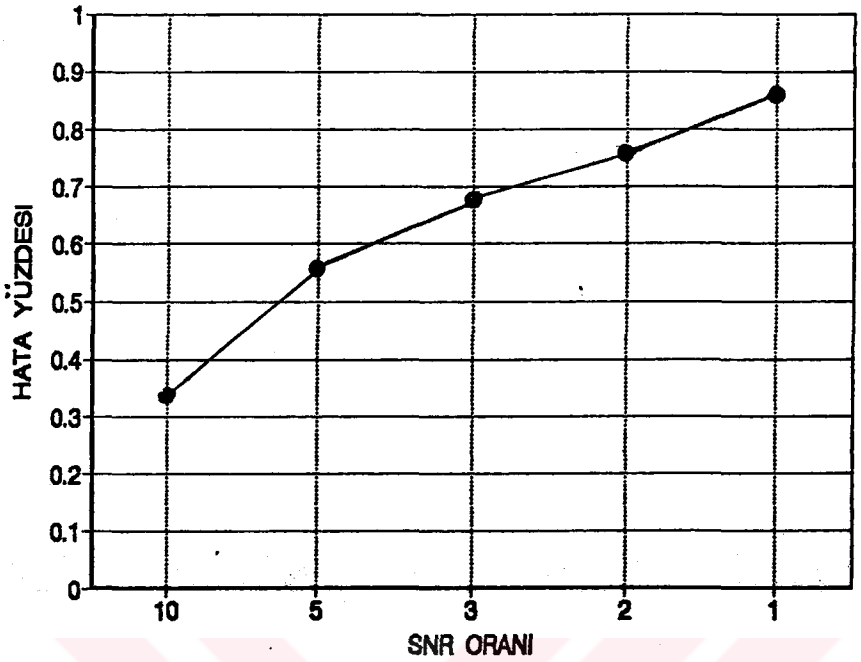
Şekil 8.5.b Yöresel İstatistik Metodu, SNR=1



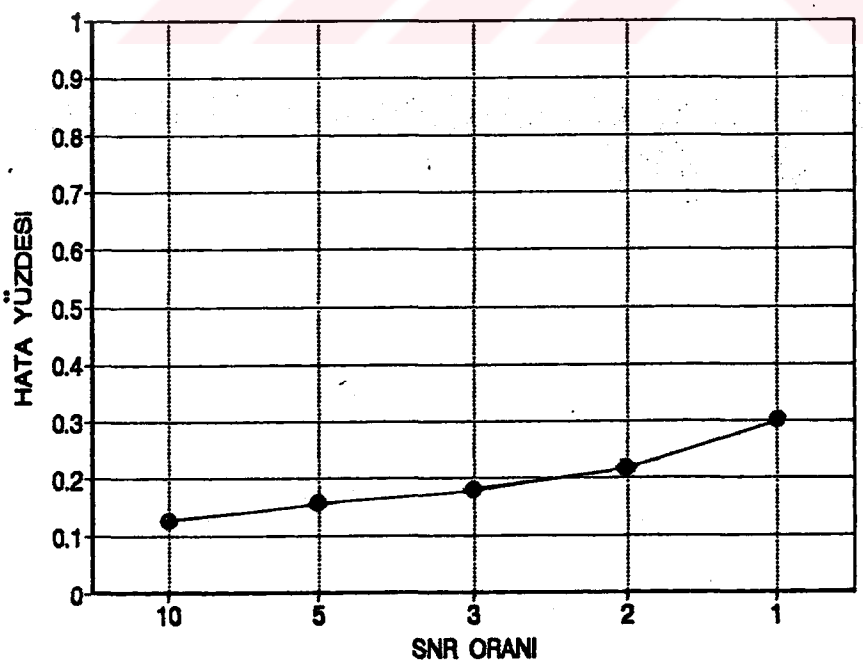
Şekil 8.7 FIR filtre kenar görüntüsü, SNR=1 , c=3.5

Şekil 8.8.a Morfolojik metod, SNR=1

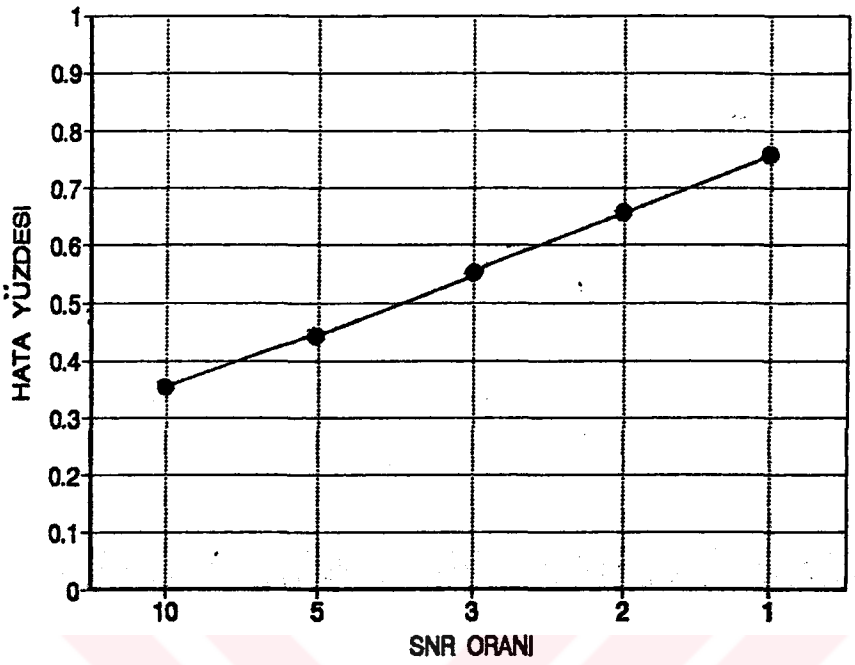
Şekil 8.8.b Morfolojik metod, SNR=1



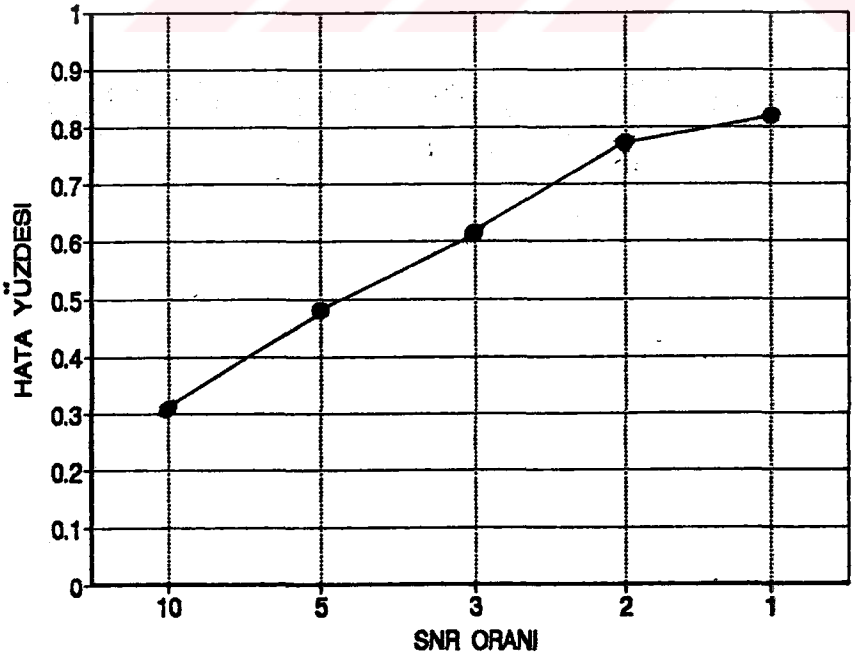
Şekil 8.9 Sobel metodu, gürültü oranı-hata yüzdesi grafiği



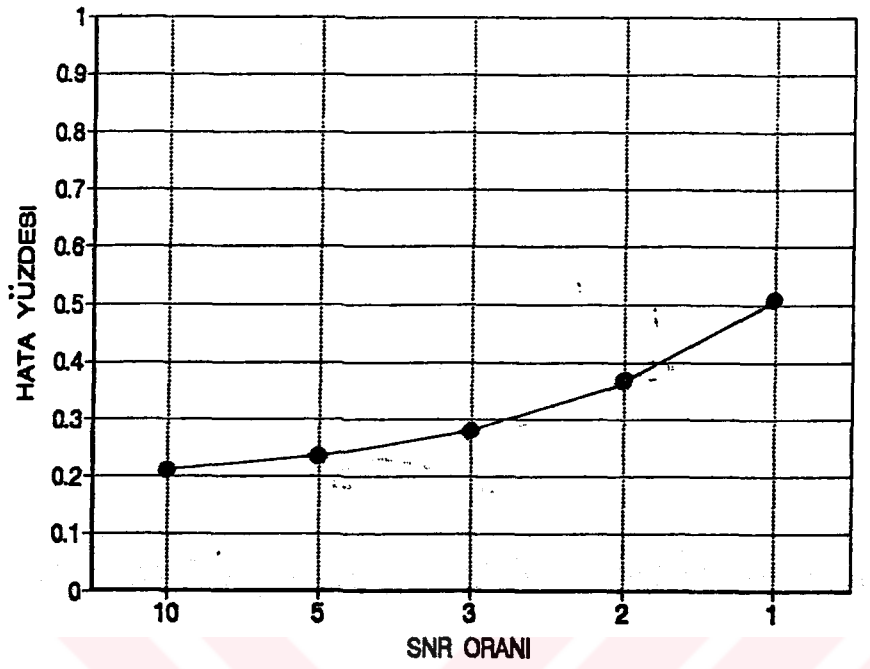
Şekil 8.10 Log filtresi gürültü oranı-hata yüzdesi grafigi



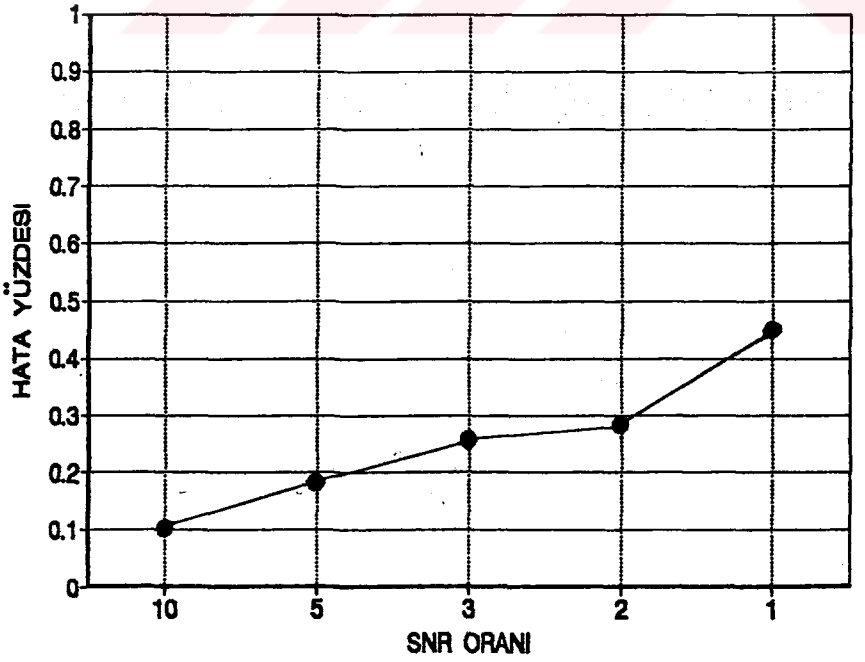
Şekil 8.13 FIR filtresi , gürültü oranı-hata yüzdesi grafiği



Şekil 8.14 Morfolojik metodu, gürültü oranı-hata yüzdesi grafiği



Şekil 8.11 Yöresel istatistik metodu gürültü oranı-hata yüzdesi grafiği



Şekil 8.12 Stokastik gradyent metodu gürültü oranı-hata yüzdesi grafiği

SONUÇ VE ÖNERİLER

Bu tezde değişik kenar tanıma algoritmaları incelendi ve her birisi için bilgisayar yazılımları hazırlandı. Bu yöntemlerdeki ortak sorunlar , kenarların gerçek yerlerinde tesbit edilememesi ve gürültülü ortamlardan çok fazla etkilenmeleridir. Bu nedenle kenar tanıma konusu hala üzerinde çalışılan bir konudur.

Bu sorunları çözebilecek bir yaklaşım , gürültüyüde hesaba katan modelleme tabanlı stokastik yaklaşım olabilir. Görüntünün kenarlarını modeller ve model parametrelerinin doğru olarak kestirebilirsek, performansı yüksek bir yöntem elde edebiliriz. Diğer bir yöntemde son yıllarda önemi gittikçe artan sinir ağlarıdır. Sinir ağlarını kullanarakta performansı yüksek bir yöntem elde edebiliriz.

KAYNAKLAR

1. Jae S.LIM ,Two Dimensinonal Signal and Image Processing ,Prentice-Hall, 1990

2. Anil K. Jain, Fundamentals of Digital Image Processing ,Prentice-hall, 1989

3. Andres Huertas, Gerard Medioni, Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks,IEEE, Trans. on Pattern Analysis and Machine Intellgence, Sept. 1986. sayfa 651-664

4. Azriel Rosenfield, Avinash C. Kak , Digital Picture Processing, Academic Press, 1976

5. Minsoo Suk, Soonho Hong, An Edge Extraction Technique for Noisy Images, Computer Vision, Graphics and Image Processing,1984,sayfa 24-45

6. J.W.Modestino, R.W.Fries, Edge Detection in Noisy Images Using Recursive Digital Filtering. Computer Graphics and Image Processing, 1977, sayfa 409-433

7. K.Sam Shanmugam, Optimal Frequency Filter for Edge Detection in Digital Picture,IEEE Transactions on Pattern Analysis and Machine Intellegence,1979, sayfa 37-49

8. Edward R.Dougherty, Charles R. Giardina, Morphological methods in Image and Signal Processing, Prentice-Hall, 1989

9. James S.J. Lee, Morphological Edge Detection , Journel of Robotics and Automation , 1987, sayfa 142-157

10. Gilbert Shaw, Local and Regional Edge Detectors: Some Comparisons. Computers Graphics and Image Processing ,1979, sayfa 135-149

Ek 1.
Bilgisayar Programlarının Açıklanması

Tezde incelenen herbir algoritma için bilgisayar programları yazılmıştır. Bilgisayar programları C programlama dilinde yazılmış ve Matematiksel işlemcili 80386 bilgisayarda koşulmuşlardır. Giriş verisi olarak, 256 gri seviyeli sayısal görüntüler kullanılmıştır. Kullanımı ;

<program ismi> <giriş veri dosyası ismi> ←——┘

Yazılan programlar aşağıdadır.

1) Gradyent.C : 3. bölümdeki gradyent metodlarının gerçeklendiği programdır.Alt programlar
2) Laplasyan.C : 3. bölümdeki laplasyan metodlarının gerçeklendiği programdır.
3) Winavera.C : Yöresel istatistik metodunun gerçeklendiği programdır.
4) Stograd.C : Stokastik gradyent metodunun gerçeklendiği programdır.
5) Edenergy.C: 6.bölümdeki FIR filtresinin gerçeklendiği programdır.
6) Morp.C : Morfolojik kenar operatörünün gerçeklendiği programdır.

Kullanılan alt programları ve işlevleri şunlardır.

void gradient(void) : Gradyent metodları
void mask(void) : Maskeleme
void threshold(void) : Eşik değerinin uygulanması
void compass(void) : Kompas metodu
signed char commask(signed char cm[3][3],int i, int j) : Kompas maskeleme
void laplacian(void) : Laplasyan metodları menusü
void lapmask(void) : Laplasyan metodu maskeleme
void variance(void) : Görüntünün varyansının bulunması
void lap_gas(void) : LoG metodu
void t_reg(unsigned char *w_p, unsigned char *ww_p) Yöresel istatistik metodu büyük pencereler ortalaması farklarının bulunması
void t_reg_m(int i,int j,unsigned char *ww_p) : Yöresel istatistik metodu küçük pencereler ortalaması farklarının bulunması
void w_mean(unsigned char *w_p) : Büyük pencerelerin ortalamasının hesaplanması
void ww_mean(unsigned char *ww_p) : Küçük pencere ortalamalarının hesaplanması
void w_var(unsigned char *w_v) : Yöresel varyansın

```
hesaplanması
void win_edge(int i, int j, unsigned char *w_p) : Yöresel
varyansın piksellere uygulanması
void grad_wnr(void) : Stokastik gradyent maskeleme
double denk_çoz(double *a) : Stokastik gradyent metodu
maske paremetrelerinin bulunması
float cor(int x,int y) : Korolasyon matrisinin bulunması
void t_morp(void)  : Morfolojik kenar operasyonu
void t_blur(void)  : Görüntünün bulanıklaştırılması
void t_eros(void)  : Erosyon işlevinin uygulanması
void t_dil(void)   : Dilasyon işlevinin uygulanması
void b_eros(void)  : Bulanık görüntünün erosyonu
void b_dil(void)   : Bulanık görüntünün dilasyonu
void display(void) : Kenar görüntüsünü ekrana verme
void misplay(void) : Görüntüyü ekrana verme
```

EK2. BILGISAYAR PROGRAMLARI

GRADYENT METODLARI

```
#include <stdio.h>
#include <math.h>
#include <tez.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>

signed char vertical[3][3] = { {-1, 0, 1 },
                               {-1, 0, 1 },
                               {-1, 0, 1 }
                             };
signed char horizantal[3][3] = { { 1,  1,  1 },
                                 { 0,  0,  0 },
                                 {-1, -1, -1 }
                               };
signed char for_ortho[3][3] = { { 0, 1 , 1},
                                {-1 ,0 , 1},
                                {-1 ,-1, 0}
                               };
signed char back_ortho[3][3] = { { 1 , 1, 0},
                                 { 1 , 0,-1},
                                 { 0 ,-1,-1}
                               };
signed char sobel_x[3][3] = { { -1, 0, 1},
                              { -2, 0, 2},
                              { -1, 0, 1}
                            };
signed char sobel_y[3][3] = { { 1, 2, 1},
                              { 0, 0, 0},
                              {-1,-2,-1}
                            };
signed char robert_x[3][3] = { { 0, 1, 0},
                               {-1, 0, 0},
                               { 0, 0, 0}
                             };
signed char robert_y[3][3] = { { 1, 0, 0},
                               { 0,-1, 0},
                               { 0, 0, 0}
                             };
signed char prewit_x[3][3] = { { -1, 0, 1},
                               { -1, 0, 1},
                               { -1, 0, 1}
                             };
signed char prewit_y[3][3] = { {-1,-1,-1},
                               { 0, 0, 0},
                               { 1, 1, 1}
                             };

signed char compass1[3][3] = { { 0, -1, 1},
                               {-1,  0, 1},
                               {-1,  1, 0} };
```

```
signed char compass2[3][3] = { { -1, 1, 0},
                               { -1, 0, 1},
                               {  0,-1, 1}
                               };

signed char compass3[3][3] = { { 1, 1, 1},
                               { 0, 0, 0},
                               {-1,-1,-1}
                               };

signed char compass4[3][3] = { { -1, 0 , 1},
                               { -1, 0 , 1},
                               { -1, 0 , 1}
                               };

main(int argc, char *argv[])
{

 int i,j,k;
 FILE *imge;
 int mts;
 char rc,nc,con,ms,ds;
 signed char wnn;
 float  wnvar,wn,error;

 if(argc!=2) { printf("  Giris digital goruntusunu yok ! \a \n ");
               printf(" <Kullanimi>  (.EXE dosyasi) (Digital goruntu dosyasi)
               exit(1);
             }
 if((imge=fopen(argv[1],"rb"))==NULL)
   {
    printf(" Goruntu dosyasi bulunamiyor ! \a \n");
    exit(1);
   }
 fread(input,MAX_X*MAX_Y,1,imge);
 fclose(imge);

 for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y;j++) input[i][j] >>=3;
 printf("Resmi gormek istiyor musunuz ? (E/H)  \n ");
 rc=getche();printf("\n");
 if((rc=='E')||(rc=='e'))
    {  misplay();
       printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
       getche();
    }

 printf(" Resme gurultu katmak istiyor musunuz ? (E/H) \n");
 nc=getche();printf("\n");
 if((nc=='E')||(nc=='e'))
   {
    printf(" Gurultunun varyansini giriniz \n");
    scanf("%f",&wnvar);
    for(i=0;i<MAX_X;i++)
      for(j=0;j<MAX_Y;j++)
        { wn=(sqrt(wnvar/0.85)*(((float)rand()/(float)RAND_MAX)-.5));
          input[i][j]+=wn;
        }
```

```
    misplay();
    printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
    getche();
   }
  printf("Medyan filtre uygulamak istiyormusunuz ? E/H \n");
  ms=getche();
  if((ms=='E')||(ms=='e'))
    {printf(" 3x3 veya 5x5 medyan filtre ?   3/5 ? ");
     scanf("%d",&mts);
     if(mts==3) { median_3();
                  misplay();
                  getche();
                }
     else if(mts==5) { median();
                       misplay();
                       getche();
                     }
    else printf("\nMedyan filtre uygulanamadi\n");
   }

  gradient();
  threshold();
  display();

}

void gradient(void)
 {
   int i,j;
   char sl,dir_sl,nondir_sl;

   do { printf("select type of operator\n");
        printf("1:directional operator\n");
        printf("2:non-directional operator\n");
        printf("3:compass operator\n");
        printf("4:quit\n");

        sl=getche()-'0';
        printf("\n");

        switch(sl)
          {
            case 1:
              printf("1:Vertical edge detector \n");
              printf("2:Horizontal edge detector\n");
              printf("3:Backward Diagonal edge detector\n");
              printf("4:Forward diagonal edge detector\n");
              printf("4:Quit\n");
              printf("enter your choice\n");
              dir_sl=getche()-'0';
              printf("\n");

              switch(dir_sl)
               {
                 case 1:
                   mask(vertical);
                   break;

                 case 2:
```

```
                    mask(horizantal);
                    break;
                  case 3:
                    mask(back_ortho);
                    break;
                  case 4:
                    mask(for_ortho);
                    break;    ;
                 }
                break;

              case 2:
                printf("1: Roberts edge detector \n");
                printf("2: Prewitt edge detector \n");
                printf("3: Sobel edge detector \n");
                printf("4: quit\n");

                printf("Enter your choice\n");
                nondir_sl=getche()-'0';
                printf("\n");

                switch(nondir_sl)
                  {
                   case 1:
                     mask(robert_x);
                     mask(robert_y);
                     break;
                   case 2:
                     mask(prewit_x);
                     mask(prewit_y);
                     break;
                   case 3:
                     mask(sobel_x);
                     mask(sobel_y);
                     break;
                   case 4:
                     exit(1);
                  }
                break;

              case 3:
                compass();
                break;
              case 4: return;
            }

     } while((sl<1)||(sl>4));

 }

void mask(signed char oprt[3][3])
  {
    int n,m,i,j;
    signed char  sum;
    for(n=1;n<(MAX_X-1);n++)
      for(m=1;m<(MAX_Y-1);m++)
        {
```

```
          for(i=0,sum=0;i<3;i++)
             for(j=0;j<3;j++)
              sum+=oprt[i][j]*input[n-1+i][m-1+j] ;
          output[n][m]+=abs(sum);
        }
  }

void threshold(void)
  {
      int i,j;
      int thres;

      printf("enter threshold value \n");
      scanf("%d",&thres);

      for(i=0;i<MAX_X;i++)
         for(j=0;j<MAX_Y;j++)
         { if(output[i][j] > (signed char)thres)  output[i][j]=1;
           else output[i][j]=0;
         }
  }

 void compass(void)
 {
 int i,j,k,l;
 signed char max1,max2;
 signed char  c1,c2,c3,c4;

 for(i=1;i<MAX_X;i++)
   for(j=1;j<MAX_Y;j++)
   {
   c1=commask(compass1,i,j);
   c2=commask(compass2,i,j);
   c3=commask(compass3,i,j);
   c4=commask(compass4,i,j);
   if(c1>c2) max1= c1;
        else  max1= c2;
   if(c3>c4) max2= c3;
        else  max2= c4;

   if(max1>max2) output[i][j]=max1;
            else output[i][j]=max2;
    }
 }

 signed char commask(signed char cm[3][3],int i,int j)
 {
   signed char com_out,com_out1,com_out2,com_out3;
   com_out1=cm[0][0]*input[i-1][j-1]+cm[0][1]*input[i-1][j]+cm[0][2]*input[i-1][
   com_out2=cm[1][0]*input[i][j-1]+cm[1][1]*input[i][j]+cm[1][2]*input[i][j+1];
   com_out3=cm[2][0]*input[i+1][j-1]+cm[2][1]*input[i+1][j]+cm[2][2]*input[i+1][
   com_out=(signed char) abs(com_out1+com_out2+com_out3);
   return com_out;
 }
```

LAPLASYAN METODU

```
#include <stdio.h>
#include <math.h>
#include <tez.h>
#include <conio.h>
#include <stdlib.h>

float *g_fil;
signed char lapla[3][3] = { { 0,  1, 0},
                            { 1, -4, 1},
                            { 0,  1, 0}
                          };

main(int argc, char *argv[])
{

 int i,j,k;
 FILE *imge,*bak_gor;
 int mts;
 char rc,nc,con,ms,ds;
 signed char wnn;
 float  wnvar,wn,error;

   if(argc!=2) { printf("  Giris digital goruntusunu yok ! \a \n ");
                 printf(" <Kullanimi>  (.EXE dosyasi) (Digital goruntu dosyasi)
                 exit(1);
               }
   if((imge=fopen(argv[1],"rb"))==NULL)
     {
      printf(" Goruntu dosyasi bulunamiyor ! \a \n");
      exit(1);
     }
   fread(input,MAX_X*MAX_Y,1,imge);

   fclose(imge);

   for(i=0;i<MAX_X;i++)
     for(j=0;j<MAX_Y;j++) input[i][j] >>=3;

   printf("Resmi gormek istiyor musunuz ? (E/H)  \n ");
   rc=getche();printf("\n");
   if((rc=='E')||(rc=='e'))
      {  misplay();
         printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
         getche();
      }
   printf(" Resme gurultu katmak istiyor musunuz ? (E/H) \n");
   nc=getche();printf("\n");
   if((nc=='E')||(nc=='e'))
     {
      printf(" Gurultunun varyansini giriniz \n");
      scanf("%f",&wnvar);
      for(i=0;i<MAX_X;i++)
         for(j=0;j<MAX_Y;j++)
            { wn=(sqrt(wnvar/0.85)*(((float)rand()/(float)RAND_MAX)-.5));
              input[i][j]+=wn;
            }
```

```
      misplay();
      printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
      getche();
     }
   printf("Medyan filtre uygulamak istiyormusunuz ? E/H \n");
   ms=getche();
   if((ms=='E')||(ms=='e'))
     {printf(" 3x3 veya 5x5 medyan filtre ?   3/5 ? ");
      scanf("%d",&mts);
      if(mts==3) { median_3();
                   misplay();
                   getche();
                  }
      else if(mts==5) { median();
                        misplay();
                        getche();
                      }
      else printf("\nMedyan filtre uygulanamadi\n");
     }
   laplacian();
   display();

}

void laplacian(void)
{
 int i,j,k,l,m,n;
 char sl;

 do{
   printf(" laplacian operatorleri \n");
   printf(" 1:laplacian maskesi \n");
   printf(" 2:Yoresel varyans testli laplacian operatoru \n");
   printf(" 3:Laplacian of Gaussion filtresi (LoG) \n");
   printf(" 4:Cikis \n");
   sl=getche()-'0';
   printf("\n");
   } while((sl>4)||(sl<1));

   switch (sl)
   {
    case 1: lapmask();
            threshold();
            break;
    case 2: lapmask();
            threshold();
            variance();
            break;
    case 3: lap_gas();
            threshold();
            break;
    case 4: break;
   }

}

void lapmask(void)
{
```

```
 int i,j;
 for(i=1;i<MAX_X;i++)
   for(j=1;j<MAX_Y;j++)
   output[i][j]=(signed char) abs(input[i][j-1]+input[i][j+1]+input[i-1][j]+inpu
}


 void variance(void)
 {
  int i,j,k,l,m,n;
  int thres;
  int mean,var;
  printf("Varyans esik degerini giriniz \n");
  scanf("%d",&thres);

  for(i=1;i<(MAX_X-1);i++)
    for(j=1;j<(MAX_Y-1);j++)
    { if(output[i][j])
        { for(k=0,var=0;k<3;k++)
            for(l=0;l<3;l++)
              {
              for(m=0,mean=0;m<3;m++)
                for(n=0;n<3;n++)  mean+=input[i-2+k+m][j-2+l+n];
              mean/=9;
              var+=(((int)input[i-1+k][j-1+l]-mean)*((int)input[i-1+k][j-1+l]-m
              }
          var/=9;
          if(abs(var) > thres) output[i][j]=1;
                          else output[i][j]=0;
        }
    }
}


void lap_gas(void)
{
 int i,j,k,l;
 int w,kat;
 float var,log_sum;

 printf("\n Gaussian filtresi varyans degerini giriniz \n");
 scanf("%f",&var);
 w=3*2.86*var;

 g_fil=malloc(sizeof(float)*(2*w-1)*(2*w-1));

 for(i=0;i<w;i++)
   for(j=0;j<w;j++)
   LOG((w-1-i),(j+w-1))=((i*i+j*j-2*M_PI*var)*exp(-(i*i+j*j)/(2*M_PI*var)))/(M_P

 for(i=0;i<w;i++)
   for(j=0;j<w;j++) LOG(i,j)=LOG(i,(2*w-2-j));
 for(i=w;i<(2*w-1);i++)
   for(j=0;j<(2*w-1);j++) LOG(i,j)=LOG((2*w-2-i),j);


 for(i=w;i<(MAX_X-w);i++)
    for(j=w;j<(MAX_Y-w);j++)
```

```
      {for(k=0,log_sum=0.0;k<(2*w-1);k++)
         for(l=0;l<(2*w-1);l++) log_sum+=(input[i-7+k][j-7+l]*LOG(k,l));
       output[i][j]=log_sum;
      }
 }

void threshold(void)
{
 int i,j;
 int thres;

 printf("Kenar goruntusu icin esik degerini giriniz \n");
 scanf("%d",&thres);

 for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y;j++)
     {
     if(abs(output[i][j]) > (signed char)thres)  output[i][j]=1;
     else output[i][j]=0;
     }
}
```

YÖRESEL İSTATİSTİK METODU

```
#include <stdio.h>
#include <math.h>
#include <tez.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>


float var_th;
main(int argc, char *argv[])
{

 int i,j,k;
 FILE *imge,*bak_gor;
 unsigned char *w_p,*ww_p,*w_v;
 int menu,mts;
 char rc,nc,con,ms,ds;
 signed char wnn;
 float  wnvar,wn,error;



   if(argc!=2) { printf("  Giris digital goruntusunu yok ! \a \n ");
                 printf(" <Kullanimi>  (.EXE dosyasi) (Digital goruntu dosyasi)
                 exit(1);
               }
   if((imge=fopen(argv[1],"rb"))==NULL)
     {
       printf(" Goruntu dosyasi bulunamiyor ! \a \n");
       exit(1);
     }
   fread(input,MAX_X*MAX_Y,1,imge);

   fclose(imge);

   for(i=0;i<MAX_X;i++)
     for(j=0;j<MAX_Y;j++) input[i][j] >>=2;

   printf("Resmi gormek istiyor musunuz ? (E/H)  \n ");
   rc=getche();printf("\n");
   if((rc=='E')||(rc=='e'))
      {  misplay();
         printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
         getche();
      }

   printf(" Resme gurultu katmak istiyor musunuz ? (E/H) \n");
   nc=getche();printf("\n");
   if((nc=='E')||(nc=='e'))
     {
       printf(" Gurultunun varyansini giriniz \n");
       scanf("%f",&wnvar);
       for(i=0;i<MAX_X;i++)
          for(j=0;j<MAX_Y;j++)
             { wn=(sqrt(wnvar/0.85)*(((float)rand()/(float)RAND_MAX)-.5));
```

```
                input[i][j]+=wn;
              }
          misplay();
          printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
          getche();
        }
      printf("Medyan filtre uygulamak istiyormusunuz ? E/H \n");
      ms=getche();
      if((ms=='E')||(ms=='e'))
        {printf(" 3x3 veya 5x5 medyan filtre ?   3/5 ? ");
         scanf("%d",&mts);
         if(mts==3) { median_3();
                      misplay();
                      getche();
                     }
         else if(mts==5) { median();
                           misplay();
                           getche();
                         }
         else printf("\nMedyan filtre uygulanamadi\n");
        }
      w_p=malloc(1*MAX_X*MAX_Y);
      ww_p=malloc(1*MAX_X*MAX_Y);
      w_v=malloc(1*MAX_X*MAX_Y);
      printf("\nYoresel varyansi giriniz  :  ");
      scanf("%f",&var_th);
      w_mean(w_p);
      ww_mean(ww_p);
      w_var(w_v);
      t_reg(w_p,ww_p);
      win_var(w_v);
      free(w_p);
      free(ww_p);
      free(w_v);
      display();


}

void w_mean(unsigned char *w_p)
{
 int i,j,k,l;
 int w_sum;
 for(i=2;i<(MAX_X-2);i+=3)
   for(j=2;j<=(MAX_Y-2);j+=3)
     {
      for(k=0,w_sum=0;k<5;k++)
        for(l=0;l<5;l++) w_sum+=input[i-2+k][j-2+l];
      WMR(i,j)=w_sum/25;
     }
}

void ww_mean(unsigned char *ww_p)
{
 int i,j,k,l;
 int ww_sum;
 for(i=1;i<(MAX_X-1);i+=1)
   for(j=1;j<=(MAX_Y-1);j+=1)
     {
      for(k=0,ww_sum=0;k<3;k++)
```

```
        for(l=0;l<3;l++) ww_sum+=input[i-1+k][j-1+l];
      WWMR(i,j)=ww_sum/9;
     }
}

void t_reg(unsigned char *w_p,unsigned char *ww_p)
{
 int i,j,k,l;
 int thres_win;
 printf("Pencereler icin esik degerini giriniz \n");
 scanf("%d",&thres_win);

 for(i=2;i<(MAX_X-2);i+=3)
   for(j=2;j<(MAX_Y-2);j+=3)
   {
    if(abs(WMR(i,j)-WMR(i,(j+3)))>thres_win)
      { if(WMR(i,j)>WMR(i,(j+3)))   t_reg_m(i,j,ww_p);
                                else   t_reg_m(i,(j+3),ww_p);
      }
    if(abs(WMR(i,j)-WMR((i+3),j))>thres_win)
      { if(WMR(i,j)>WMR((i+3),j))  t_reg_m(i,j,ww_p);
                                else  t_reg_m((i+3),j,ww_p);
       }
   }
}

void t_reg_m(int i,int j,unsigned char *ww_p)
{
    if(abs(WWMR((i-1),(j-1))-WWMR((i-1),(j+1)))>=1)
      { if(WWMR((i-1),(j-1))>WWMR((i-1),(j+1)))  win_edge((i-1),(j-1),ww_p);
                                          else   win_edge((i-1),(j+1),ww_p);
      }
    if(abs(WWMR((i-1),(j-1))-WWMR((i+1),(j-1)))>=1)
      { if(WWMR((i-1),(j-1))>WWMR((i+1),(j-1)))  win_edge((i-1),(j-1),ww_p);
                                          else  win_edge((i+1),(j-1),ww_p);
       }
    if(abs(WWMR((i-1),(j-1))-WWMR((i+1),(j+1)))>=1)
      { if(WWMR((i-1),(j-1))>WWMR((i+1),(j+1)))  win_edge((i-1),(j-1),ww_p);
                                          else   win_edge((i+1),(j+1),ww_p);
      }
    if(abs(WWMR((i+1),(j-1))-WWMR((i+1),(j+1)))>=1)
      { if(WWMR((i+1),(j-1))>WWMR((i+1),(j+1)))  win_edge((i+1),(j-1),ww_p);
                                          else  win_edge((i+1),(j+1),ww_p);
       }
    if(abs(WWMR((i+1),(j-1))-WWMR((i-1),(j+1)))>=1)
      { if(WWMR((i+1),(j-1))>WWMR((i-1),(j+1)))  win_edge((i+1),(j-1),ww_p);
                                          else   win_edge((i-1),(j+1),ww_p);
      }
    if(abs(WWMR((i-1),(j+1))-WWMR((i+1),(j+1)))>=1)
      { if(WWMR((i-1),(j+1))>WWMR((i+1),(j+1)))  win_edge((i-1),(j+1),ww_p);
                                          else  win_edge((i+1),(j+1),ww_p);
       }


}

void win_edge(int i,int j,unsigned char *w_p)
{
```

```
 int k,l;
 for(k=0;k<3;k++)
   for (l=0;l<3;l++)  if(input[i-1+k][j-1+l]>=WMR(i,j)) output[i-1+k][j-1+l]=1;
}

void w_var(unsigned char *w_v)
{

 int i,j,m,n,k,l;
 float mean,var;

 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++)
     {
     for(k=-1,var=0;k<2;k++)
       for(l=-1;l<2;l++)
         {
          for(m=-1,mean=0;m<2;m++)
            for(n=-1;n<2;n++)  mean+=(float)(input[i+k+m][j+l+n]);
          mean/=9;
          var+=(((int)input[i+k][j+l]-mean)*((int)input[i+k][j+l]-mean));
         }
     WVAR(i,j)=var/9;
     }
 }

void win_var(unsigned char *w_v)
{
 int i,j;
 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++) { if(output[i][j]==1)
                                {if(WVAR(i,j)<var_th) output[i][j]=0;}
                            }

}
```

STOKASTİK GRADYENT METODU

```
#include <stdio.h>
#include <math.h>
#include <tez.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>
#include <alloc.h>
float pv,ph,rff;
unsigned int top;
long int meanim,irff,irfflo,irffol;
float rfflo,rffol,co,snr;
float gv[5][5],gh[5][5],gsum,gkat;
double   *a,d;
int mean,err_sum,o_det, o_det_m,mts;
char rc,nc,con,ms,ds;
signed char wnn;
float  wnvar,wn,error;

main(int argc, char *argv[])
{


 FILE *imge,*bak_trec;
 int i,j,k,l,m,n;


 if(argc!=2) { printf("  Giris digital goruntusunu yok ! \a \n ");
               printf(" <Kullanimi>  (.EXE dosyasi) (Digital goruntu dosyasi)
               exit(1);
             }
 if((imge=fopen(argv[1],"rb"))==NULL)
   {
     printf(" Goruntu dosyasi bulunamiyor ! \a \n");
     exit(1);
   }
 fread(input,MAX_X*MAX_Y,1,imge);

 fclose(imge);

 for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y;j++) input[i][j] >>=3;

 printf("Resmi gormek istiyor musunuz ? (E/H)  \n ");
 rc=getche();printf("\n");
 if((rc=='E')||(rc=='e'))
    {  misplay();
       printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
       getche();
    }
 top=MAX_X*MAX_Y;

 printf(" Resme gurultu katmak istiyor musunuz ? (E/H) \n");
 nc=getche();printf("\n");
 if((nc=='E')||(nc=='e'))
   {
   printf("Gurultunun varyansini giriniz /n");
   scanf("%f",&wnvar);
```

```
   for(i=0;i<MAX_X;i++)
     for(j=0;j<MAX_Y;j++)
        { wn=(sqrt(wnvar/0.85)*(((float)rand()/(float)RAND_MAX)-.5));
          input[i][j]+=wn;
         }
   misplay();
   printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
   getche();
   }

   printf("Medyan filtre uygulamak istiyormusunuz ? E/H \n");
   ms=getche();
   if((ms=='E')||(ms=='e'))
     {printf(" 3x3 veya 5x5 medyan filtre ?   3/5 ? ");
      scanf("%d",&mts);
      if(mts==3) { median_3();
                   misplay();
                   getche();
                  }
      else if(mts==5) { median();
                        misplay();
                        getche();
                       }
      else printf("\nMedyan filtre uygulanamadi\n");
     }

meanim=0;
for(i=0;i<MAX_X;i++)
  for(j=0;j<MAX_Y;j++) meanim+=input[i][j];
mean=meanim/top;

irff=0;
for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y;j++) irff+=((input[i][j]-mean)*(input[i][j]-mean));
rff=irff/top;

irffol=0;
for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y-1;j++) irffol+=((input[i][j+1]-mean)*(input[i][j]-mean));
rffol=irffol/top;
ph=rffol/rff;

irfflo=0;
for(i=0;i<MAX_X-1;i++)
   for(j=0;j<MAX_Y;j++) irfflo+=((input[i+1][j]-mean)*(input[i][j]-mean));
rfflo=irfflo/top;
pv=rfflo/rff;

printf("pv: %f , ph: %f \n",pv,ph);
a= malloc(sizeof(double)*(2*6+1)*6);
printf("\nSNR oranini giriniz ");
scanf("%f",&snr);
for(i=0;i<6;i++)
 for(j=0;j<=2*6;j++)
   {
   if(j<6 ) { if(i==j) A(i,j)=(1/snr)+cor((j/3-i/3),(i%3-j%3));
                  else A(i,j)=cor((j/3-i/3),(i%3-j%3));
            }
```

```
    else if(j<5 || j<12) { if(i==(j-6)) A(i,j)=1;
                           else A(i,j)=0;
                         }
    else if(j==2*6) A(i,j)=cor((-i/3),(i%3));
    }

 denk_coz(a);

 gv[2][1]=A(0,12);
 gv[1][1]=gv[3][1]=A(1,12);
 gv[0][1]=gv[4][1]=A(2,12);
 gv[2][0]=A(3,12);
 gv[1][0]=gv[3][0]=A(4,12);
 gv[0][0]=gv[4][0]=A(5,12);
 free(a);
 gsum=0.0;
 for(i=0;i<5;i++)
   for(j=0;j<2;j++) gsum+=gv[i][j];
 gsum=(1-gsum)/10;

 for(i=0;i<5;i++)
   for(j=0;j<2;j++) gv[i][j]+=gsum;

 for(i=0;i<5;i++)
    {
    if(j=2) gv[i][j]=0.0;
    for(j=3;j<5;j++) gv[i][j]=-gv[i][4-j];
    }

 gkat=1/gv[2][1];
 for(i=0;i<5;i++)
   for(j=0;j<5;j++) gv[i][j]*=gkat;

 for(i=0;i<5;i++)
   for(j=0;j<5;j++)  gh[i][j]=gv[j][i];

 grad_wnr();
 threshold();
 display();

}

float cor(int x,int y)
{
   int i,j;
   float acor;
   acor=pow(pv,abs(x))*pow(ph,abs(y));
   return acor;
}


double denk_coz(double *a)
 {
  register   i,j,k;
  double det = 1.0;

  for(k=0;k<6;k++) { if(A(k,k)==0.0)  { sat_deg(a,k);
                                        det*= -1.0;
```

```
                                    }
                    for(i=0;i<6;i++) { if(i==k) continue;
                                       for(j=k+1;j<=2*6;j++)
                                         A(i,j)-=A(k,j)*A(i,k)/A(k,k);
                                     }
                  }
  for(k=0;k<6;det*=A(k,k),k++)
    for(j=6;j<=2*6;j++) A(k,j)/=A(k,k);

  return(det);
 }

sat_deg(double *a,int k)
 {
   register i,j;
   double   tmp;

   for(i=k+1;i<6 && A(i,k)==0.0;i++);

   if(i==6) exit(1);

   for(j=k;j<=2*6;j++) { tmp    = A(k,j);
                         A(k,j) = A(i,j);
                         A(i,j) = tmp;
                       }
 }

void grad_wnr(void)
 {
   int i,j,k,l;
   float maskv,maskh,maskw;

   for(i=2;i<MAX_X-2;i++)
     for(j=2;j<MAX_Y-2;j++)
       {
       maskv=maskh=0.0;
       for(k=0;k<5;k++)
         for(l=0;l<5;l++)
          {
          maskh+=input[i-2+k][j-2+l]*gh[k][l];
          maskv+=input[i-2+k][j-2+l]*gv[k][l];
          }
       maskw=(fabs(maskv)+fabs(maskh))/2;
       output[i][j]=(signed char)maskw;
       }
 }

void threshold(void)
{
      int i,j;
      int thres;

      printf("Esik degerini giriniz \n");
      scanf("%d",&thres);

      for(i=0;i<MAX_X;i++)
         for(j=0;j<MAX_Y;j++)
         {
```

```
        if(output[i][j] > thres)  output[i][j]=1;
        else output[i][j]=0;
        }
}

 void median(void)
 {
 int i,j,m,n;
 unsigned char temp;
 unsigned char median[9];


 for(i=2;i<MAX_X-2;i++)
   for(j=2;j<MAX_Y-2;j++)
     {
      median[0]=input[i-2][j];
      median[1]=input[i-1][j];
      median[2]=input[i+1][j];
      median[3]=input[i+2][j];
      median[4]=input[i][j];
      median[5]=input[i][j-2];
      median[6]=input[i][j-1];
      median[7]=input[i][j+1];
      median[8]=input[i][j+2];
      for(m=0;m<9;m++)
        for(n=8;n>m;n--)
        {if(median[n-1]<median[n])
           {
           temp=median[n-1];
           median[n-1]=median[n];
           median[n]=temp;
           }
         }
    input[i][j]=median[4];
    }

}
void median_3(void)
{
int i,j,m,n;
unsigned char temp;
unsigned char median[5];


for(i=2;i<MAX_X-2;i++)
  for(j=2;j<MAX_Y-2;j++)
    {
     median[0]=input[i-1][j];
     median[1]=input[i][j];
     median[2]=input[i+1][j];
     median[3]=input[i][j-1];
     median[4]=input[i][j+1];
     for(m=0;m<5;m++)
       for(n=4;n>m;n--)
       {if(median[n-1]<median[n])
          {
          temp=median[n-1];
          median[n-1]=median[n];
```

```
            median[n]=temp;
            }
        }
    input[i][j]=median[2];
    }

}
```

FIR FİLTRE METODU

```
#include <stdio.h>
#include <math.h>
#include <tfft.h>
#include <conio.h>
#include <stdlib.h>

float  wre,wim,t_re,t_im,t2_re,t2_im,ts_re,ts_im;
float xre[MAX_Y],xim[MAX_Y],h_re[MAX_Y],h_im[MAX_Y],xdft_re[MAX_Y],xdft_im[MAX_Y
float f_kat,out_re[MAX_Y],out_im[MAX_Y],out_fil[MAX_Y];


main(int argc, char *argv[])
{

int i,j,k,s,n,r;
FILE *imge,*th_fft;
int menu,M;
long int xl,xl2;
int hd,thres;
float rc,wo,omega,bi,kat,outsq;
if(argc!=2) { printf("input image intensity file name is missing\n");
            exit(1);
            }
if((imge=fopen(argv[1],"rb"))==NULL)
  {
  printf("image file cannot be open\n");
  exit(1);
  }

fread(input,MAX_X*MAX_Y,1,imge);

fclose(imge);

for(i=0;i<MAX_X;i++)
  for(j=0;j<MAX_Y;j++) input[i][j] >>=3;

/*  misplay(); getch();*/

printf("Band genisligi-cozunurluk sabitini giriniz, c sabiti \n");
scanf("%f",&rc);
bi=3.14;
omega=2*rc/bi ;
wo=(2*M_PI)/(2*MAX_Y+1);
printf("filtre sabitini giriniz  \n");
scanf("%f",&kat);
printf("rc:%f bi:%f omega:%f \n wo:%f kat:%f \n",rc,bi,omega,wo,kat );

for(n=0;n<MAX_Y;n++)
  {
  f_kat=(-rc*rc*rc*n*n*wo*wo)/(2*omega*omega*omega*omega);
  if((float)n*wo<omega)  h_re[n]= kat*n*n*wo*wo*exp(f_kat);
            else  h_re[n]=0.0 ;
  h_im[n]=0.0 ;
  }
```

```
for(r=0;r<MAX_X;r++)
 {
  for(j=0;j<MAX_X;j++)
    {
    xre[j]=input[j][r];
    xim[j]=0.0 ;
    }
  xre[255]=xim[255]=0.0;

  M=(log10(MAX_Y)/log10(2));
  for(i=0;i<M;i++)
    {
    xl=1<<(M-i);
    xl2=xl>>1 ;
    for(j=0;j<xl2;j++)
      {
      wre=cos(j*M_PI/xl2) ;
      wim=-sin(j*M_PI/xl2) ;
      for(k=j;k<MAX_Y;k+=xl)
        {
        t_re=xre[k]+xre[k+xl2];
        t_im=xim[k]+xim[k+xl2];
        t2_re=xre[k]-xre[k+xl2];
        t2_im=xim[k]-xim[k+xl2];
        xre[k+xl2]=(t2_re*wre)-(t2_im*wim);
        xim[k+xl2]=(t2_re*wim)+(t2_im*wre);
        xre[k]=t_re;
        xim[k]=t_im;
        }
      }
    }

/*  FFT nin sirasinin degistirilmesi   */

   j=1;
   for(i=1;i<MAX_Y;i++)
     {
     if(i<j) { ts_re=xre[j-1];
               ts_im=xim[j-1];
               xre[j-1]=xre[i-1];
               xim[j-1]=xim[i-1];
               xre[i-1]=ts_re;
               xim[i-1]=ts_im;
             }
     hd=MAX_Y/2;
     while (hd<j) { j-=hd;
                    hd/=2;
                  }
     j+=hd;
     }

/* G[i][j] nin bulunmasi  */

  for(j=0;j<MAX_Y;j++)
    {
     out_re[j]=xre[j]*h_re[j]-xim[j]*h_im[j];
     out_im[j]=xre[j]*h_im[j]+xim[j]*h_re[j];
    }

  for(j=0;j<MAX_Y;j++)
```

```
    { xre[j]=out_re[j];
      xim[j]=out_im[j];
    }

/* inverse FFT'nin alinmasi */
  for(i=0;i<=M;i++)
    {
    xl=pow(2,(M-i));
    xl2=xl>>1 ;
    for(j=0;j<xl2;j++)
      {
      wre=cos(j*M_PI/xl2) ;
      wim=sin(j*M_PI/xl2) ;
      for(k=j;k<MAX_Y;k+=xl)
         {
         t_re=xre[k]+xre[k+xl2];
         t_im=xim[k]+xim[k+xl2];
         t2_re=xre[k]-xre[k+xl2];
         t2_im=xim[k]-xim[k+xl2];
         xre[k+xl2]=(t2_re*wre)-(t2_im*wim);
         xim[k+xl2]=(t2_re*wim)+(t2_im*wre);
         xre[k]=t_re;
         xim[k]=t_im;
         }
      }
    }

/*IFFT nin sirasinin degistirilmesi*/

    j=1;
    for(i=1;i<MAX_Y;i++)
      {
      if(i<j) { ts_re=xre[j-1];
                ts_im=xim[j-1];
                xre[j-1]=xre[i-1];
                xim[j-1]=xim[i-1];
                xre[i-1]=ts_re;
                xim[i-1]=ts_im;
              }
      hd=MAX_Y/2;
      while (hd<j) { j-=hd;
                     hd/=2;
                   }
      j+=hd;
      }

  for(i=0;i<MAX_Y;i++)
     { xre[i]/=MAX_Y;
       xim[i]/=MAX_Y;
       outsq=sqrt(xre[i]*xre[i]+xim[i]*xim[i]);
       out_tem[r][i]=outsq;

     }
 }
 for(i=0;i<MAX_X;i++)
   for(j=0;j<MAX_Y;j++)  output[i][j]=out_tem[j][i];

/* horizantal dogrultudaki kenarlar   */

for(r=0;r<MAX_X;r++)
```

```
{
 for(j=0;j<MAX_Y;j++)
   {
   xre[j]=input[r][j];
   xim[j]=0.0 ;
   }

 M=(log10(MAX_Y)/log10(2));
 for(i=0;i<M;i++)
   {
   xl=1<<(M-i);
   xl2=xl>>1 ;
   for(j=0;j<xl2;j++)
     {
     wre=cos(j*M_PI/xl2) ;
     wim=-sin(j*M_PI/xl2) ;
     for(k=j;k<MAX_Y;k+=xl)
       {
       t_re=xre[k]+xre[k+xl2];
       t_im=xim[k]+xim[k+xl2];
       t2_re=xre[k]-xre[k+xl2];
       t2_im=xim[k]-xim[k+xl2];
       xre[k+xl2]=(t2_re*wre)-(t2_im*wim);
       xim[k+xl2]=(t2_re*wim)+(t2_im*wre);
       xre[k]=t_re;
       xim[k]=t_im;
       }
     }
   }

/*  FFT nin sirasinin degistirilmesi   */

   j=1;
   for(i=1;i<MAX_Y;i++)
     {
     if(i<j) { ts_re=xre[j-1];
               ts_im=xim[j-1];
               xre[j-1]=xre[i-1];
               xim[j-1]=xim[i-1];
               xre[i-1]=ts_re;
               xim[i-1]=ts_im;
             }
     hd=MAX_Y/2;
     while (hd<j) { j-=hd;
                    hd/=2;
                  }
     j+=hd;
     }

/* G[i][j] nin bulunmasi  */

  for(j=0;j<MAX_Y;j++)
    {
     out_re[j]=xre[j]*h_re[j]-xim[j]*h_im[j];
     out_im[j]=xre[j]*h_im[j]+xim[j]*h_re[j];
    }

  for(j=0;j<MAX_Y;j++)
```

```
    { xre[j]=out_re[j];
      xim[j]=out_im[j];
    }

/* inverse FFT'nin alinmasi */
  for(i=0;i<=M;i++)
    {
    xl=pow(2,(M-i));
    xl2=xl>>1 ;
    for(j=0;j<xl2;j++)
      {
      wre=cos(j*M_PI/xl2) ;
      wim=sin(j*M_PI/xl2) ;
      for(k=j;k<MAX_Y;k+=xl)
         {
         t_re=xre[k]+xre[k+xl2];
         t_im=xim[k]+xim[k+xl2];
         t2_re=xre[k]-xre[k+xl2];
         t2_im=xim[k]-xim[k+xl2];
         xre[k+xl2]=(t2_re*wre)-(t2_im*wim);
         xim[k+xl2]=(t2_re*wim)+(t2_im*wre);
         xre[k]=t_re;
         xim[k]=t_im;
         }
      }
    }

/*IFFT nin sirasinin degistirilmesi*/

    j=1;
    for(i=1;i<MAX_Y;i++)
      {
      if(i<j) { ts_re=xre[j-1];
                ts_im=xim[j-1];
                xre[j-1]=xre[i-1];
                xim[j-1]=xim[i-1];
                xre[i-1]=ts_re;
                xim[i-1]=ts_im;
              }
      hd=MAX_Y/2;
      while (hd<j) { j-=hd;
                     hd/=2;
                   }
      j+=hd;
      }

  for(i=0;i<MAX_Y;i++)
     { xre[i]/=MAX_Y;
       xim[i]/=MAX_Y;
       outsq=sqrt(xre[i]*xre[i]+xim[i]*xim[i]);
       out_tem[r][i]=outsq;

     }
}
printf("\n");
getche();

for(i=0;i<MAX_X;i++)
```

```
    for(j=0;j<MAX_Y;j++)  output[i][j]+=out_tem[i][j];

  th_fft=fopen("fft.the","wb");
  for(i=0;i<MAX_X;i++)
    for(j=0;j<MAX_Y;j++) fwrite(output[i][j],1,1,th_fft);
  fclose(th_fft);

  printf("enter threshold value");
  scanf("%d",&thres);
  for(i=0;i<MAX_X;i++)
    for(j=0;j<MAX_Y;j++)
       { if(output[i][j]>thres) output[i][j]=1;
                                else output[i][j]=0;
       }

  display();

}
```

MORFOLOJİK METOD

```
#include <stdio.h>
#include <math.h>
#include <tez.h>
#include <conio.h>
#include <stdlib.h>
#include <alloc.h>


main(int argc, char *argv[])
{

   int i,j,k;
   FILE *imge;
   int menu;
   char rc,nc,con,ms,ds;
   signed char wnn;
   float  wnvar,wn,error;


   if(argc!=2) { printf("  Giris digital goruntusunu yok ! \a \n ");
                 printf(" <Kullanimi>  (.EXE dosyasi) (Digital goruntu dosyasi)
                 exit(1);
               }
   if((imge=fopen(argv[1],"rb"))==NULL)
     {
       printf(" Goruntu dosyasi bulunamiyor ! \a \n");
       exit(1);
     }
   fread(input,MAX_X*MAX_Y,1,imge);

   fclose(imge);

   for(i=0;i<MAX_X;i++)
     for(j=0;j<MAX_Y;j++) input[i][j] >>=3;

   printf("Resmi gormek istiyor musunuz ? (E/H)  \n ");
   rc=getche();printf("\n");
   if((rc=='E')||(rc=='e'))
      {  misplay();
         printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
         getche();
      }

   printf(" Resme gurultu katmak istiyor musunuz ? (E/H) \n");
   nc=getche();printf("\n");
   if((nc=='E')||(nc=='e'))
     { printf(" Gurultunun varyansini giriniz \n");
       scanf("%f",&wnvar);
       for(i=0;i<MAX_X;i++)
          for(j=0;j<MAX_Y;j++)
             { wn=(sqrt(wnvar/0.85)*(((float)rand()/(float)RAND_MAX)-.5));
               input[i][j]+=wn;
             }
        misplay();
        printf(" Devam etmek icin herhangi bir tusa basiniz. \n");
        getche();
```

```
    }
  printf("Medyan filtre uygulamak istiyormusunuz ? E/H \n");
  ms=getche();
  if((ms=='E')||(ms=='e'))
    {printf(" 3x3 veya 5x5 medyan filtre ?   3/5 ? ");
     scanf("%d",&mts);
     if(mts==3) { median_3();
                  misplay();
                  getche();
                }
     else if(mts==5) { median();
                       misplay();
                       getche();
                     }
    }

  t_blur();
  t_dil();
  t_eros();
  b_eros();
  b_dil();
  t_morp();
  threshold();
  display();

}

void t_blur(void)
{
int i,j,k,l;
int blur_sum;
for(i=1;i<(MAX_X-1);i++)
  for(j=1;j<(MAX_Y-1);j++)
  { for(k=0,blur_sum=0;k<3;k++)
      for(l=0;l<3;l++)
      blur_sum+=input[i-1+k][j-1+l];
    output[i][j]=blur_sum/9;
  }
for(i=1;i<(MAX_X-1);i++)
  for(j=1;j<(MAX_X-1);j++) input[i][j]=output[i][j];
}

void t_dil(void)
{
int i,j,k,l;
int max;

for(i=2;i<(MAX_X-2);i++)
  for(j=2;j<(MAX_Y-2);j++)
  { max=input[i-2][j-2];
    for(k=0;k<5;k++)
      for(l=0;l<5;l++)
      if(input[i-2+k][j-2+l]>max) max=input[i-2+k][j-2+l];
    dil[i][j]=max;
  }
}

void t_eros(void)
{
 int i,j,k,l;
```

```
 int min;

 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++)
   { min=input[i-2][j-2];
     for(k=0;k<5;k++)
       for(l=0;l<5;l++)
       if(input[i-2+k][j-2+l]<min) min=input[i-2+k][j-2+l];
     eros[i][j]=min;
   }
 }

void b_eros(void)
{
 int i,j;
 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++)  eros[i][j]=input[i][j]-eros[i][j];
}

void b_dil(void)
{
 int i,j;
 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++)  dil[i][j]=dil[i][j]-input[i][j];
}


void t_morp(void)
{
 int i,j;
 for(i=2;i<(MAX_X-2);i++)
   for(j=2;j<(MAX_Y-2);j++)
   { if(dil[i][j]>eros[i][j]) output[i][j]=eros[i][j];
                         else output[i][j]=dil[i][j];
   }
}


void threshold(void)
  {
      int i,j;
      int thres;

      printf("enter threshold value \n");
      scanf("%d",&thres);

      for(i=0;i<MAX_X;i++)
         for(j=0;j<MAX_Y;j++)
         {
         if(output[i][j] > (signed char)thres)  output[i][j]=1;
         else output[i][j]=0;
         }
 }
```

```
include <tez.h>

#define  MAX_X   254
#define  MAX_Y   256
#define  C_TETA  1.2732396
#define  BUYUME 1.1
#define LOG(i,j)  (*(g_fil+(i)*2*w-(i)+(j)))
#define WMR(i,j)  (*(w_p+(i)*MAX_Y+(j)))
#define WWMR(i,j) (*(ww_p+(i)*MAX_Y+(j)))
#define WVAR(i,j) (*(w_v+(i)*MAX_Y+(j)))
#define A(i,j)    (*(a+(i)*2*6+(i)+(j)))

extern unsigned char input[MAX_X][MAX_Y];
extern signed   char output[MAX_X][MAX_Y];
extern signed   char out_n[MAX_X][MAX_Y];
extern signed   char out_y[MAX_X][MAX_Y];
extern signed   char eros[MAX_X][MAX_Y];
extern signed   char dil[MAX_X][MAX_Y];


void gradient(void);
void mask(signed char oprt[3][3]);
void magnitude(signed char in1[MAX_X][MAX_Y]);
void threshold(void);
void thinning(void);
void magnitude_2(signed char in1[MAX_X][MAX_Y],signed char in2[MAX_X][MAX_Y]);
void thresh(void);
void dynamic(void);
void laplacian(void);
void lapmask(void);
void lap_gas(void);
void variance(void);
void grad_dir(signed char in1[MAX_X][MAX_Y], signed char in2[MAX_X][MAX_Y]);
void median(void);
void median_3(void);
void regional(void);
void compass(void);
signed char commask(signed char cm[3][3],int i, int j);
void t_reg(unsigned char *w_p,unsigned char *ww_p);
void t_reg_m(int i,int j,unsigned char *ww_p);
void w_mean(unsigned char *w_p);
void ww_mean(unsigned char *ww_p);
void w_var(unsigned char *w_v);
void win_var(unsigned char *w_v);
void win_srch(int i,int j,unsigned char *w_p);
void win_edge(int i,int j,unsigned char *w_p);
void w_thin(int i,int j,unsigned char *w_p);
void t_blur(void);
void t_eros(void);
void t_dil(void);
void b_eros(void);
void b_dil(void);
void t_morp(void);
void grad_wnr(void);
double denk_coz(double *a);
sat_deg(double *a,int k);
float cor(int x,int y);
```

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

ÖZGEÇMİŞ

21.6.1967 yılında Isparta'nın Senirkent ilçesinde doğdum. İlk , orta ve lise öğrenimimi Balıkesir'de yaptım. 1984 yılında Balıkesir Lisesinden mezun olarak aynı yıl Boğaziçi Üniversitesi Elektrik ve Elektronik Muhendisliği bölümüne girdim. 1989 yılında mezun oldum. 1990 yılında İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme A.B.D. yüksek lisans öğrenimine başladım. Şu anda Bekoteknik Sanayii A.Ş. inde araştırma mühendisi olarak çalışmaktayım.

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ