DİJİTAL GÖRÜNTÜ İŞLEME SİSTEMİNDE GÖRÜNTÜ DÜZGÜNLEŞTİRİLMESİ VE GÖRÜNTÜ ZENGİNLEŞTİRİLMESİ

Abdulselam YILMAZ

Yüksek Lisans Tezi

Elektronik Anabilim Dalı

1992

DİJİTAL GÖRÜNTÜ İŞLEME SİSTEMİNDE
GÖRÜNTÜ DÜZGÜNLEŞTİRİLMESİ ve GÖRÜNTÜ ZENGİNLEŞTİRİLMESİ

Abdulselam YILMAZ

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Lisansüstü Yönetmenliği Uyarınca

Elektronik Anabilim Dalında

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır.

Danışman : Doç. Dr. Hamdi ATMACA

Şubat-1992

Abdulselam YILMAZ'ın YÜKSEK LİSANS tezi olarak hazırladığı " DİJİTAL GÖRÜNTÜ İŞLEME SİSTEMİNDE GÖRÜNTÜ DÜZGÜNLEŞTİRİLMESİ ve GÖRÜNTÜ ZENGİNLEŞTİRİLMESİ " başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Üye : Doç.Dr. Hamdi ATMACA

Üye : Prof.Dr. Atalay BARKANA

Üye : Yrd.Doç.Dr. Abdurrahman KARAMANCIOĞLU

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..../..../1992 gün ve 31.4-6... sayılı kararıyla onaylanmıştır.

Prof.Dr. Rüstem KAYA

Enstitü Müdürü

İÇINDEKILER

İÇİNDEKİLER (devam)

EKLER LİSTESİ

ÖZET

1920'lerde basit bir şekilde yapılmakta olan Dijital Görüntü İşleme (DGİ) Sistemi, günümüze kadar bilgisayarların gelişmesine paralel olarak bir çok aşamadan geçmiştir.

Özellikle 1970'lerden sonra büyük bir gelişme gösteren DGİ sistemi, tıp, TV, animasyon, uydu haberleşmeleri gibi alanlarda sık sık kullanılmaktadır. Prensip olarak bir kaynaktan alınan analog bilgi matris şeklinde gösterilen dijital bilgiye çevrilmektedir. Bu dijital bilgi değişik işlemlerden geçirilerek görüntü olarak ekrana gönderilmektedir.

Bilgisayarlarla beraber gelişen ve günümüzde yaygın olarak kullanılan bilgisayar destekli görüntüleme sistemleri gittikçe artmaktadır. Bu uygulamalar, animasyon ve tıp bilimlerinin gelişmesinde büyük bir rol oynamıştır.

Bu projenin amacı, bilgisayarlı tomografi cihazlarında doktorların hasta izleme ve rapor yazmak için kullandıkları ve ikinci konsol adı verilen cihazın yerine IBM uyumlu bir PC nin yerleştirilmesidir. Bunun için IBM PC'de çalışacak şekilde ve yukarda bahsedilen cihazın özelliklerini fonksiyonel olarak içeren bir uygulama programı yazılmıştır. Bu program ileride eklenecek yeni uygulamalar için modüler biçimde meydana getirilmiştir. Bu projede yazılan bir ana işletim programı ve üç adet uygulama programı, dijital görüntüleme sistemlerinde kullanılan tüm özelliklere sahiptir.

SUMMARY

The Digital Image Processing has gained important role in various medical applications along with the development of speedy and larged memory personal computers. Especially the techniques of image processing have found applications in picture animation, telecommunications, television and medical diagnosis.

In this project, a PC was inserted as a second console for easy use of computer aided tomography in a hospital. For this IBM compatible PC a main and other supporting programs were developed in order to track the medical pictures on CRT for diagnostic purposes. These modular programs contained many flexible facilities such as enlargement, rotation, enhancement and smoothing for obtaining suitable pictures on high resolution colour monitor. In addition, a data histogram colouring and three dimentional pictures can easily be seen on the screen of the PC.

Therefore, this software was developed by means of these programs in the form of Turbo Pascal and Assembly language under MS-DOS operation system.

TEŞEKKÜR

Bu projenin tasarımlanması ve gerçekleşmesi konusunda benden yardımlarını esirgemeyen Sayın Doç.Dr. Hamdi ATMACA'ya bu konuda ilk çalışmalarımı teşvik eden Sayın Rıfat EDİZKAN'a ODTU Bilgisyar Mühendisliği araştırma görevlilerine, Atatürk Üniversitesi Tıp Fakültesi Radyoloji Ana Bilim Dalı Öğr.Üyesi Sayın Yrd.Doç.Dr. Adnan Okur'a ve bana lisans eğitimim yıllarında programlama dillerini öğreten Sayın Murat YILDIRIM'a teşekkürü bir borç bilirim.

BÖLÜM 1

GİRİŞ

1.1 Giriş

İlk dijital görüntüleme yöntemleri 1920'lerde gazeteler için geliştirilen ve Londra ile NewYork arasında kodlanarak gönderilen resim bilgileridir. Bununla birlikte resim kalitesinin kötü olması ışıklandırma seviyesinin de (brightness level) dikkate alınmasını gerektirmiştir. İlk olarak beş seviye oluşturulmuş, 1929 yılında bu 15 seviyeye çıkarılmıştır. Bu olay 1964 yılına kadar devam etmiştir.

1964 yılından günümüze kadar Dijital Görüntü İşleme (DGİ) sistemleri, X-ışınları ile tıpta, ışık dağılımı ve renk kullanılarak baskı işlemlerinde, TV reklamları ve çizgi film animasyonlarında kullanılacak seviyeye getirilmiştir.

1.2 Dijital Görüntüleme İfadeleri

Bu çalışmada, monokrom görüntü veya basit görüntü, (simple image) iki boyutlu ışık yoğunluk fonksiyonu (two-dimensional light intensity function) f(x,y) ile modellenmiştir. Burada x ve y uzaysal koordinatlar olup ve f(x,y) değeri, diğer bir ifade ile, görüntü üzerinde bulunan (x,y) noktasının grilik seviyesidir.

Bir dijital görüntü (x,y) uzaysal koordinatlarla f(x,y) grilik seviyesinden oluşmaktadır. Dijital görüntü, bir matris ile gösterilip, elemanları o noktanın grilik seviyesini vermektedir. Bu elemanlara görüntü elemanı, resim elemanı, pixel veya pel denir.

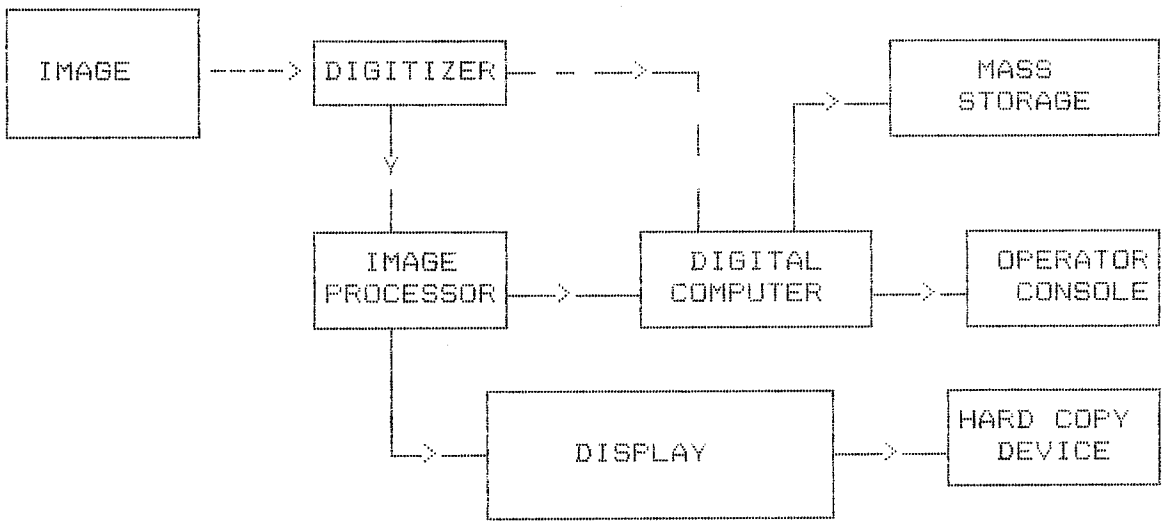
1.3 Dijital Görüntüleme Sistem Elemanları

1.3.1 Digitizer

Digitizer, herhangı bir görüntüyü dijital bilgisayarların girisine kabul edilebilecek bir sekilde sayısal ifadeye çeviren bir cihazdır. En çok kullanılan digitizer cihazları, mikrodensitometreler, hareketli nokta tarayıcıları (flying spot scanner), görüntü parcalayıcılar (image dissector) ve fotoduyarlı ayrılmıs durum dizinleri (photosensitive split-state array). İlk iki cihaz da görülebilen veya fotoğrafik resimler ve diğerlerinde ise resim kayıtları kullanılmaktadır(1).

1.3.2 Görüntü İşleyicileri

Dijital görüntü isleyicileri, görüntüleme sisteminin temelini olusturmaktadır. Bir dijital görüntü isleyicisi donanım olarak, dört ana modül olan görüntü toplama, saklama, düsük seviyeli (hızlı) islemci ve ekrandan meydana gelmektedir(Sekil 1.1).



Sekil 1.1 Dijital görüntüleme sistem elemanları.

Görüntü toplama birimi, giris olarak anolog TV sinyallerini alıp dijital değerlere cevirir. Bu dijital değerlerde uzaysal koordinatlar ve o noktanın grilik seviyesi bulunmaktadır.

Saklama birimi 'frame buffer' olarak adlandırılır ve dijital görüntünün saklandığı hafızadır. Hafıza adresleri görüntü biriminin taramasına göre yapılır.

Islemci birimi, aritmetik ve lojik islemlerini hızlı yapabilmesi için "low-level" lisanda programlanır. Bu birim Aritmetik-Lojik ünitesi (ALU) olarak adlandırılır. Bu bölümde alınan anolog sinyaller dijital bilgiye cevrilerek hafızaya yazılır.

1.3.3 Dijital Bilgisayar

Dijital bilgisayar, genel amaçlı ve programlanması kolay olan birer ara birimdir. Bilgisayar sistemleri, olusacak görüntünün boyutlarına göre değişmektedir; bu boyutlar, görüntü buffer büyüklüğü, hız ve aynı anda kac tane görüntü sayısına sahip olmasına bağlıdır. Bu boyutların iyi olması istenirken sistem maliyetinin düşük olmasına dikkat edilir.

1.3.4 Depolama Birimleri

Her pixeli sekiz bit (256 renk veya tonlama) olan bir 512 * 512 pixellik dijital görüntü, 0.25 megabyte yer kaplar. Bu da genel amaçlı dijital görüntüleme sisteminde veri saklama problemlerini meydana getirir. Bu problemleri cözmek için aşağıdaki depolama birimleri kullanılabilmektedir.

i. Magnetik Diskler

ii. Magnetik Teypler

iii. Optik Diskler

Bir magnetik diskin kapasitesi yaklasık olarak 700 megabyte olup yukarıdaki özellikte 2800 görüntü saklama sayısına sahiptir. Magnetik teyplerin yüksek yoğunluklu olanları tercih edilmektedir. Optik diskler, lazerle okuma ve yazma prensibine dayanan cihazlardır. Bir optik diskin kapasitesi dört gigabyte ve yaklasık 16000 görüntü saklama sayısına sahiptir. Optik disklerin dezavantajı ise, yazılmıs kayıtların silinememeleridir. Sony firmasının en son gelistirdiği ve halen deneme asamasında bulunan video kaset kaydediciler bir saatlik bir video kasetine 6000'in üzerinde görüntü saklama sayısına sahip olduğu bilinmektedir(2,3,4).

1.3.5 Ekran ve Kayıt Birimi

Modern görüntüleme sistemlerinde prensip olarak monokrom ve renkli televizyon monitorları kullanılmaktadır. Bu monitorlar, görüntü isleyicisinin cıkısı olarak görev yaparlar. Aynı zamanda bu cıkıs sinyali, kağıt üzerine görüntü almak için görüntü kayıt birimine bağlanabilir. Bu fonksiyona da "Hard Copy" denir. Diğer bir görüntü birimi de CRT (Cathode Ray Tube) dir.

CRT sistemlerinde, görüntü dizininde bulunan her bir elemanın yatay ve dikey pozisyonlarını ve o noktada bulunan görüntü elemanın değerini voltaja cevirerek elektron bombardımanı yaparak iki boyutlu görüntü cıkısı elde edilir. Grilik seviyesi gelen voltaja göre elektron bombardımanı tarafından sağlanır. Yazıcı cihazları düsük yoğunluklu görüntüleme sistemlerinde kullanılır. Grilik seviyesi, kağıt üzerine

değişik alfanümerik karakterlerin basılması ile elde edilir. Genelde bu özellik, düşük yoğunluklu olduğundan kullanılmaz.

BÖLÜM 2

BİLGİSAYARLI TOMOGRAFİ

2.1 Bilgisayarlı Tomografi ve Kullanım Alanları

Bilgisayarlı tomografi cihazları, ilk olarak 1970'li yıllarda geliştirilmiştir. 1980 yılından sonra mikrobilgisayarlı sistemler tasarlanmıştır. Bu cihazlar tıp alanında yaygın olarak kullanılmakta olup ülkemize 1984 yılından sonra girmeye başlamıştır.

Tomografi cihazı, hasta çevresinde dönebilen bir X-RAY tüpü ile bu tüpün tam karşısında hastadan geçen X-RAY ışınlarını toplayan Xe-dedektöründen (Xenon gazı ile dolu dedektör) ibarettir. Xe-dedektörden alınan anolog bilgi, hızlı A/D çeviricilerden geçirilip dijital bilgi olarak ana bilgisayara gönderilir. Bu dijital bilgi, operatör tarafından tanımlanan özelliklere göre hızlı matematiksel işlemcilerden geçirildikten sonra görüntü olarak ekrana ve görüntü saklama birimine aktarılır. Bu cihazlarda oluşturulan görüntü, hasta üzerinden alınan dijital bilgiler daha önceden alınmış olan standart olarak kullanılan su bilgileriyle karşılaştırılarak elde edilir. Bundan dolayı bu cihazlarda yaklaşık olarak her altı ayda bir su ile kalibrasyon yapılması gerekmektedir(2,4,5).

Bilgisayarlı tomografi cihazı, elektronik olarak çok karmaşık bir yapıya sahiptir. Bunun sebebi, yapılan matematiksel ve görüntüleme işlemlerin çok hızlı olması(yedi saniyede bir görüntü elde edilmesi gibi) ve mekanik kontrollerin fazlalığıdır. Bundan dolayı matematiksel ve görüntüleme işlemleri yazılım yolu ile değil, donanım yoluyla

yapılmaktadır(2). Bu cihaza ait ana bölümler aşağıda sıralanmış ve şekil 2.1'de gösterilmiştir.

GANTRY : X-RAY tüpü, Xe-dedektör ve hızlı A/D çeviricilerin içinde bulunduğu kısımdır. Hasta bu cihazın içerisine yerleştirilir.

HASTA YATAĞI : Hastanın, hazırlanıp ve görüntülenmesi istenilen bölgesine göre gantry içerisine yerleştirmek için yatırıldığı kısımdır.

ANA BİLGİSAYAR : Hastadan alınmış dijital bilgileri işleyen ve diğer birimleri kontrol altında tutan kısımdır.

X-RAY KONTROL : Bilgisayardan aldığı komutlar doğrultusunda X-RAY tüpününün akım ve gerilimini kontrol eden kısımdır.

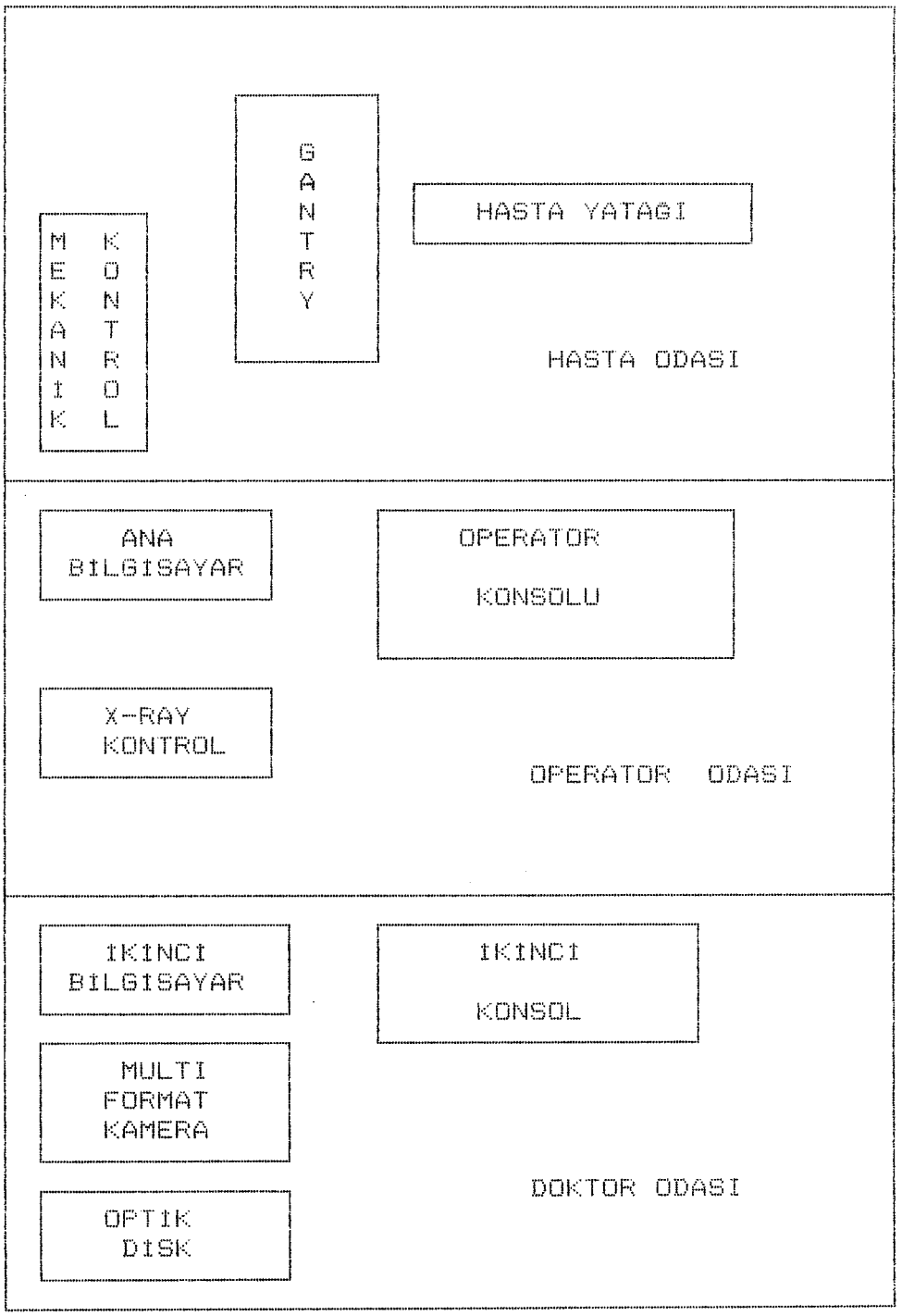
MEKANİKSEL KONTROL : Cihazdaki tüm mekanik hareketleri kontrol eden kısımdır.

OPERATÖR KONSOLU : Kullanıcının dışarıdan girdiği bilgileri (tarama bilgileri, görüntü bilgileri, hasta bilgileri gibi) içeren kısımdır. Doğrudan ana bilgisayara bağlıdır.

MULTI FORMAT KAMERA : Ekranda bulunan görüntüyü filme almak için kullanılan kısımdır. Direkt olarak görüntü ekranına bağlıdır.

İKİNCİ KONSOL ve BİLGİSAYAR : Doktorların hasta izleme ve rapor yazmak için kullandıkları ve isteğe bağlı olan kısım olup doğrudan ana bilgisayara bağlanır.

OPTİK DİSK : Hastalardan alınmış görüntülerin arşivlenmesi için kullanılan kısımdır. Bu cihaz da isteğe bağlı olarak kullanılır.

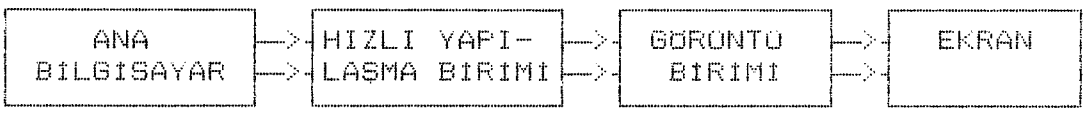


Şekil 2.1. Bilgisayarlı tomografi cihazına ait ana bölümler ve yerleşimi.

Bilgisayarlı tomografi cihazlarının kullanılması ile tıp alanında büyük gelişmeler meydana gelmiştir. Özellikle girişimsel tomografi olarak adlandırılan yöntemle, hasta üzerinde ameliyat gibi tıbbi müdahale yapılmadan teşhis koyma ve değerlendirme yapılabilmektedir.

2.2 Görüntü İşleme Birimi

Bilgisayarlı tomografi cihazlarında görüntü işleme birimi donanım olarak iki ana modülden oluşmaktadır. Bu modüller şekil 2.2'de gösterilmiştir.



Şekil 2.2 Bilgisayarlı tomografi cihazına ait görüntüleme birimi

HIZLI YAPILAŞMA BİRİMİ: Bu birimde ana bilgisayardan alınan dijital görüntü bilgileri matris notasyonuna çevrilip, filitrelenek görüntü birimine gönderilir.

GÖRÜNTÜ BİRİMİ: Bu birimde, görüntü kullanıcının istediği özelliklere göre oluşturulup ekrana gönderilir. Pencere ayarları ve özel görüntüleme işlemlerini de bu birim gerçekleştirir.

Bu birimlerde kullanılan elektronik parçalar özel olarak üretilmiş olup herhangi bir bilgi verilmemektedir. Yazılım olarak assembler dili kullanılmaktadır(4,6).

BÖLÜM 3

DIJITAL GÖRÜNTÜNÜN TEMEL KURALLARI

3.1 Görüntü Modeli

Iki boyutlu ışık şiddeti fonksiyonu f(x,y) ile gösterilen görüntü, uzaysal koordinatların (x ve y'nin) verdiği noktanın grilik seviyesidir. Işık enerji formu olan f(x,y), sıfır ve sonsuz açık aralığında bir ifadedir:

$$0 < f(x,y) < \pm\infty$$

Görüntüler, cisimden yansıyan ışık miktarını içerir ve iki elemanla karakterize edilir. Birinci eleman görüntü üzerinden yayılan ışık miktarı olup ikinci eleman görüntü üzerinden yansıyan ışık miktarıdır. Bu elemanlara sırasıyla aydınlatma ve yansıma elemanları denir ve i(x,y) ve r(x,y) ile gösterilir. Işık şiddeti fonksiyonu f(x,y), bu iki terimin çarpımı olarak ifade edilir:

$$f(x,y) = i(x,y) * r(x,y)$$

burada

$$0 < i(x,y) < \infty$$

ve

$$0 < r(x,y) < 1 \quad \text{dir.}$$

Son eşitlik, yansımanın 0 (toplam yutma) ile 1 (toplam yansıma) açık aralığında olduğunu gösterir. i(x,y) değeri ışık kaynağına göre hesaplanırken r(x,y) değeri cismin karakteristiğine göre hesaplanmaktadır. (x,y) koordinatlarındaki grilik seviyesi, $L_{min} < f(x,y) < L_{max}$ aralığındadır. Teoride, L_{min} pozitif ve L_{max} sonlu değerlerdir. Pratikte $L_{min} = i_{min} * r_{min}$ ve $L_{max} = i_{max} * r_{max}$ olup, normal ışık şartlarında $L_{min} = 0.0005$ ve $L_{max} = 100$ olarak alınır. $[L_{min}, L_{max}]$

aralığı grilik ölçeği olarak adlandırılır. Genelde bu aralık [0,L] olarak kabul edilir. I=0 iken siyah, I=L iken beyaz ölçek oluşur. Siyah ile beyaz arasındaki tüm ara değerler değişen gri tonlamalarını verir.

3.2 Örnekleme ve Basamaklandırma

Bir dijital görüntü NxN boyutlarında olan bir matris şeklinde düşünülebilir.

$$F = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,N-1) \end{bmatrix}$$

Bu eşitliğin sağ tarafı dijital görüntü ifadesi olup her bir eleman, görüntü elemanı, resim elemanı, pixel veya pel olarak tanımlanır. Ekranda oluşacak görüntünün matris boyutu N ve grilik seviyesi G ile gösterilip, bir pixeli oluşturan bit sayısı ile ilişkileri:

$$N = 2^n$$

$$G = 2^n$$

şeklindedir. Burada n ekranda bir pixeli oluşturan bit sayısıdır. Görüntü saklama kapasitesi b ile gösterilir ve b= N * N * n şeklinde hesaplanır.Değişik N ve n değerlerine göre toplam görüntü kapasitesi Tablo 3.1 ve Tablo 3.2 'de gösterilmiştir.

Örneğin 128 x 128 boyutlarında ve 64 grilik seviyesine sahip bir görüntü için 98304 bitlik bir hafıza birimi gerek-

mektedir. Görüntü ayırma gücü (resolution) N ve n değerine bağlı olarak değişmektedir. N ve n değerinin artması görüntü kalitesini artırır. Görüntülemede kullanılan minumum değerler 256 * 256 pixel ve 64 grilik seviyeden oluşmaktadır. Genel olarak 512 * 512 pixel ve n=8 (8=256 grilik seviyesi) kullanılmaktadır.

Tablo 3.1 Değişik N ve n değerlerine göre b değerleri.

(bit olarak)

N	n	2	3	4	5	6	7	8
32		2048	3072	4096	5120	6144	7168	8192
64		8192	12288	16384	20480	24576	28672	32768
128		32768	49152	65536	81920	98304	114688	131072
256		131072	196608	262144	327680	393216	458752	524288
512		524288	786432	1048576	1310720	1572864	1835008	2097152

Tablo 3.2 Değişik N ve n değerlerine göre b değerleri.

(byte olarak)

N	n	2	3	4	5	6	7	8
32		256	512	512	1024	1024	1024	1024
64		1024	2048	2048	4096	4096	4096	4096
128		4096	8192	8192	16384	16384	16384	16384
256		16384	32768	32768	65536	65536	65536	65536
512		65536	131072	131072	262144	262144	262144	262144

3.3 Pixeller Arası İlişkiler

Bu kısımda pixeller arasındaki ilişkiler açıklanmakta-

dır. Görüntü f(x,y) ve pixeller ise p ve q ile gösterilecektir.

3.3.1 Pixellerin Komşuluğu

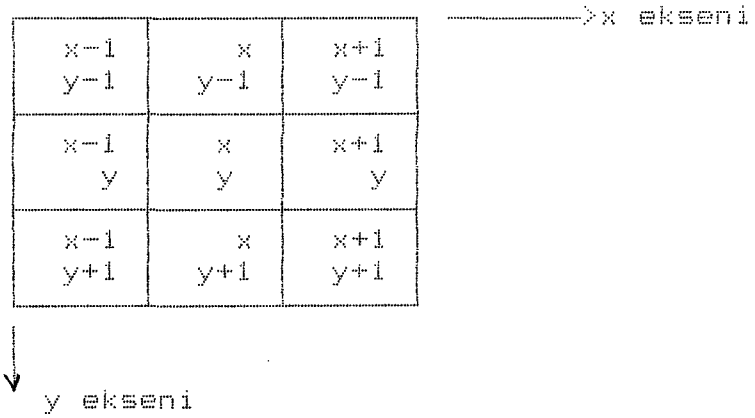
Her hangi bir (x,y) noktasında p ile gösterilen bir pixelin dört yatay ve dikey komşuluğu vardır. Bunlar

(x+1,y),(x-1,y),(x,y+1),(x,y-1) dir.

Bu pixel setine p'nin 4'lü komşuluğu denir ve $N_4(p)$ ile gösterilir. Yatay ve dikey komşuluğun dışında bulunan ve köşelerde bulunan pixellere köşegen (diagonal) komşuluk denir ve $N_D(p)$ ile gösterilir. Bunlar

(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1) olmaktadır.

Diyagonal, yatay ve dikey komşulukların tamamına p'nin 8'li komşuluğu denir ve $N_8(p)$ ile gösterilir. Bu komşuluklar Şekil 3.1'de verilmiştir.



Şekil 3.1 (x,y) noktasındaki bir pixelin tüm komşulukları.

3.3.2 Konnektivity

Konnektivity, pixeller arasında, alt görüntü bölgelerinin ayırımında kullanılan önemli bir kavramdır. Bir görüntü

üzerinde birbirine yakın pixel değerlerin toplandığı alt görüntüye konnektiviti denir. İki pixel arasındaki bağlantı pixellerin komşuluğuna göre hesaplanır ve V ile gösterilir. Bu kavram kullanılarak, görüntü üzerinde oluşabilecek hatalar giderilebilmektedir. Söyleki bir pixel ile bu pixelin komşuluğunda bulunan diğer pixellerin grilik seviyelerinin birbirine yakın değerlerde olması gerekmektedir.

Grilik seviye değerleri V, konnektivity tanımına göre set edilir. Örneğin, pixellerin değeri 59,60,61 ile verilmiş ise V={59,60,61} olur. Üç tip konnektivity vardır:

a) 4'lü konnektivity: Eğer $q \in N_4(p)$ ise p ve q pixelleri V ile 4'lü bağlıdır.

b) 8'li konnektivity: Eğer $q \in N_8(p)$ ise p ve q pixelleri V ile 8'li bağlıdır.

c) m konnektivity (mixed konnektivity):

i. Eğer $q \in N_4(p)$ ise

ii. Eğer $q \in N_D(p)$ ve $N_4(p)$ $N_4(q)$ boş ise

p ve q pixelleri V ile m'li bağlıdır.

3.3.3 Mesafe Ölçümleri

Koordinatları (x,y),(s,t) ve (u,v) ile verilen p,q ve z pixellerinin mesafe fonksiyonu D ile gösterilir. Mesafe fonksiyonu aşağıdaki şartları sağlar.

a) $D(p,q) > 0$ $(D(p,q)=0$ yalnız ve yalnız $p=q)$

b) $D(p,q) = D(q,p)$

c) $D(p,z) \leq D(p,q) + D(q,z)$

İki pixel p ve q arasındaki Euclidean mesafesi,

$D_e(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$ eşitliği ile tanımlanır.

İki pixel p ve q arasındaki D_4 mesafesi (City-Blok mesafesi)

$D_4(p,q) = |x-s| + |y-t|$ eşitliği ile tanımlanır.

Örneğin $D_4 < 2$ ise sabit mesafe ifadesini pixellerin yerleşimi (F matrisi içerisinden alınmış bir alt grup) aşağıdaki şekilde olduğu varsayılarak:

```
          2
        2 1 2
      2 1 0 1 2
        2 1 2
          2
```

p'nin dörtlü komşuluğundan dolayı $D_4 = 1$ bulunur.

İki pixel p ve q arasındaki D_8 mesafesi (Chessboard mesafesi) $D_8(p,q) = \max(|x-s|, |y-t|)$ ile tanımlanır.

Örneğin $D_8 < 2$ ise sabit mesafe ifadesi pixellerin yerleşimi aşağıdaki şekilde olduğu kabul edilirse,

```
2 2 2 2 2
2 1 1 1 2
2 1 0 1 2     D8 = 1 ve (x,y)'nin 8'li
2 1 1 1 2          komşuluğu bulunur.
2 2 2 2 2
```

3.3.4 Aritmetik / Lojik İşlemler

Görüntüleme sisteminin tüm bölümlerinde pixeller arasında aritmetik ve lojik işlemleri sık sık kullanılmaktadır. p ve q gibi iki pixelin arasındaki aritmetik işlemler şunlardır:

Toplama : p + q

Çıkarma : p − q

Carpma : p * q

Bölme : p / q

Lojik islemler ise:

AND : p AND q veya p * q

OR : p OR q veya p + q

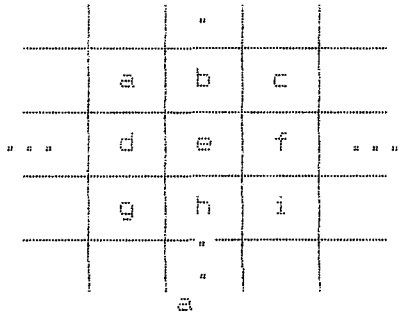
COMPLEMENT : NOT(q) veya NOT(p) olur.

Görüntülemede aritmetik ve lojik islemler için iki yol vardır. Biricisi pixel-pixel, ikincisi ise komşuluğa bağlı islemlerdir. Komşuluk islemleri formüle edilebilir ve maskeleme islemi olarak da adlandırılır. (Maskeleme olarak pencereleme ve filitreleme gibi islemler sayılabilir.) Bu islemlerle ilgili komşuluk ve maskeleme katsayıları Şekil 3.2 de verilmiştir. Bu ifade;

p=w1*a + w2*b + w3*c + w4*d + w5*e + w6*f + w7*g+w8*h +w9*i

olup burada p, e'nin maskelenmiş değeridir. Eğer W_i = 1/9 alınırsa (i = 1,2,....,9)

p = (1/9) * (a + b + c + d + e + f + g + h + i) olur.



w1	w2	w3
w4	w5	w6
w7	w8	w9

b

Şekil 3.2 Merkezi 'e' olan 3 x 3 'lük bölgenin komşulukları (a) ve maskeleme katsayıları (b)

3.4 Görüntüleme Geometrisi

Bu kısımda, görüntü rotasyonu, ölçekleme ve öteleme yön-

temleri anlatılacaktır. Tüm dönüşümler üç boyutlu (3D) olarak kartezyen sisteminde tanımlınacaktır.

3.4.1 Öteleme

(x,y,z) koordinatlarındaki bir noktayı başka bir noktaya (xO,yO,zO) değerlerine dönüşümü için

$$X1 = X + XO$$

$$Y1 = Y + YO$$

$$Z1 = Z + ZO$$

olur. Burada (X1,Y1,Z1) noktası yeni noktanın koordinatları olup matris formunda tanımlanırsa,

$$\begin{bmatrix} X1 \\ Y1 \\ Z1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & XO \\ 0 & 1 & 0 & YO \\ 0 & 0 & 1 & ZO \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

veya

$$\begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & XO \\ 0 & 1 & 0 & YO \\ 0 & 0 & 1 & ZO \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

formunda yazılabilir ve kompakt olarak V1 = A x V seklinde gösterilebilir. Burada

A: 4 x 4'lük transformasyon matrisi,

V: orijinal koordinatları içeren kolon vektörü ve

V1: dönüşüm noktasının koordinatlarını içeren kolon vektörüdür. Bu vektörler

$$V = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad V1 = \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \quad \text{ve} \quad T = \begin{bmatrix} 1 & 0 & 0 & XO \\ 0 & 1 & 0 & YO \\ 0 & 0 & 1 & ZO \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

olduğundan dönüşüm işlemi Vl = T x V ile yapılır.

3.4.2 Ölçekleme

Ölçekleme faktörleri Sx, Sy, Sz olan X, Y, Z noktasının transformasyonu

Xs = Sx * X

Ys = Sy * Y

Zs = Sz * Z ile tanımlanmaktadır.

Burada (Xs,Ys,Zs) noktası (X,Y,Z) noktasının ölçekleme yapılmış halidir. Bu dönüşüm kompakt olarak Vs = S * V şeklinde yazılabilir.

Sonuçta ölçekleme transformasyon matrisi

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ şeklinde olur.

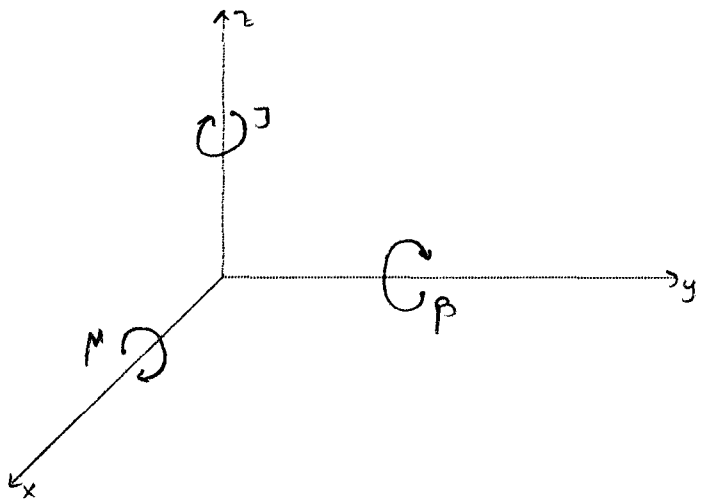
3.4.3 Rotasyon

Üç boyutlu rotasyon dönüşümü komplex bir yapıya sahiptir. Z ekseninde ve rotasyon açısı J olan bir noktanın transformasyonu

$$R_J = \begin{bmatrix} CosJ & SinJ & 0 & 0 \\ -SinJ & CosJ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklinde yazılır. Burada rotasyon açısı J, Z ekseninde bulunan noktanın orijine göre saat yönündeki ölçümü alınır ve Se-

kil 3.3'de gösterilmiştir.



Şekil 3.3 Koordinat eksenlerine göre dönüş yönleri.

X ekseninde ve rotasyon açısı μ olan bir noktanın transfor-
masyonu:

$$R_{\mu} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}\mu & \text{Sin}\mu & 0 \\ 0 & -\text{Sin}\mu & \text{Cos}\mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ile

Y ekseninde ve rotasyon açısı β olan bir noktanın transfor-
masyonu ise

$$R_{\beta} = \begin{bmatrix} \text{Cos}\beta & 0 & -\text{Sin}\beta & 0 \\ 0 & 1 & 0 & 0 \\ \text{Sin}\beta & 0 & \text{Cos}\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklinde yazılır.

3.4.4 Ters Rotasyon Transformasyonu

Tüm transformasyonlar, 4 x 4'lük transformasyon matrisi ile tanımlanmıştır. Örneğin Z ekseninde bulunan V noktasının öteleme, ölçekleme ve rotasyonu

$$Vx = R_J(S(TV))$$

$$= AV \quad \text{ifadesi ile verilir.}$$

Burada A, 4 x 4'lük transformasyon matrisi ve $A = R_J ST$'dir. Ters transformasyon matrisi aşağıda verilen şekildedir.

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -XO \\ 0 & 1 & 0 & -YO \\ 0 & 0 & 1 & -ZO \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ve benzer şekilde ters rotasyon matrisi ise

$$R_J^{-1} = \begin{bmatrix} \text{Cos}(-J) & \text{Sin}(-J) & 0 & 0 \\ -\text{Sin}(-J) & \text{Cos}(-J) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

olarak yazılabilir. Bu matrislerin komplex olması sebebiyle çözümlerde ve işlemlerde uygun nümerik teknikler kullanılmaktadır.

BÖLÜM 4

GÖRÜNTÜ ZENGİNLEŞTİRİLMESİ ve DÜZGÜNLEŞTİRİLMESİ

4.1 Giriş

Zenginleştirme teknikleri, özel uygulamalar için, orijinal görüntü üzerinde kabul edilebilir şekilde yapılan değişik işlemlerdir. Buradaki özel uygulamalar, uygulama alanına göre değişmektedir. Örneğin X-RAY kullanılarak elde edilen görüntüleri, değişik açılardan (doku yapısı veya kemik yapısına göre) görme imkanı sağlanabilir. Bu bölümde anlatılacak olan konular frekans alanı (frequency domain) veya uzaysal alan (spatial domain) yöntemleri göz önüne alınarak anlatılacaktır.

4.1.1 Uzaysal Alan Yöntemleri

Uzaysal alan yöntemleri direkt olarak, pixeller arası ilişkiler kullanılarak işlem görmeyi esas alır. Uzaysal alanda görüntü fonksiyonu

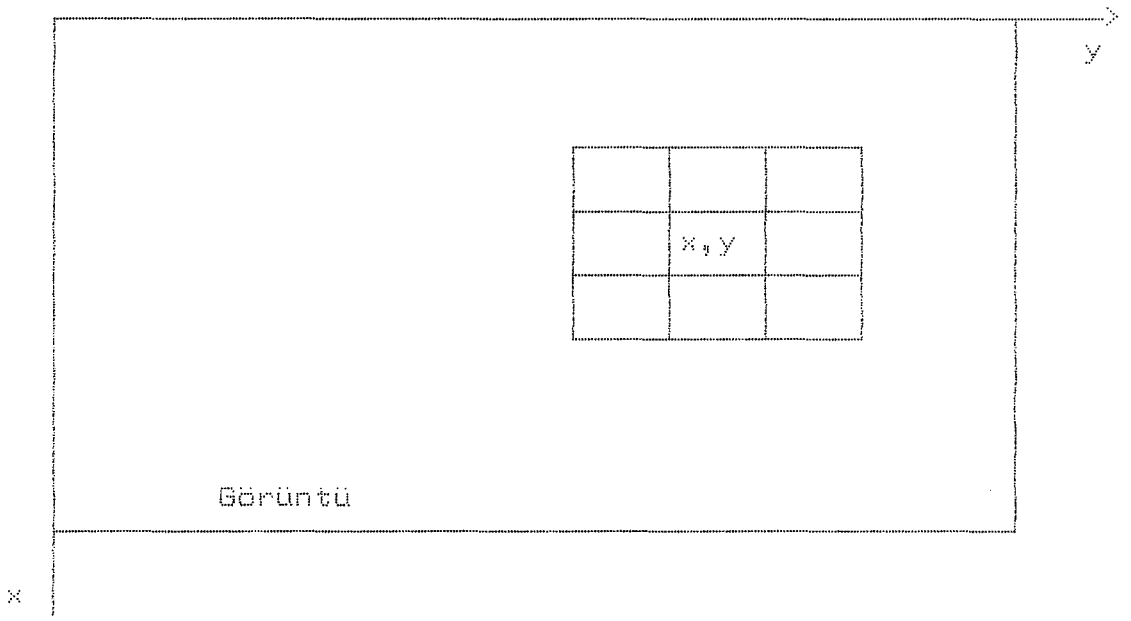
$g(x,y) = T[f(x,y)]$ ile tanımlanır.

Burada $f(x,y)$ giriş görüntüsü, $g(x,y)$ işlem görmüş görüntü ve T, (x,y) noktasına bağlı olan f'in operatörüdür. Şekil 4.1'de görüldüğü gibi (x,y) noktasının komşuluğu, merkezi (x,y) noktası olan karesel veya dikdörtgensel alt görüntüdür. En basit T formu, 1 x 1 boyutlarında olan komşuluktur. Bu durumda g, çıkış fonksiyonu yalnızca f'nin (x,y) noktasındaki değerine bağlı ve T, grilik seviyesi transformasyon fonksiyonu olur ve bu form

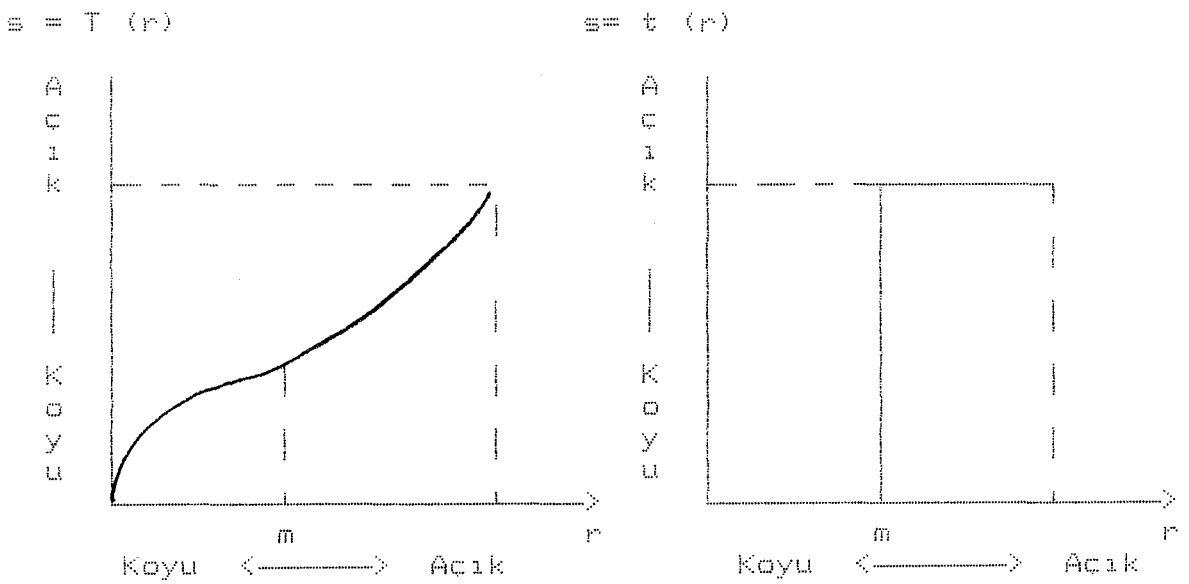
$s = T(r)$ şeklinde ifade edilir.

burada, notasyonu basitleştirmek amacıyla r ve s değişkenleri

Şekil 4.2'de gösterildiği gibi her hangi bir (x,y) noktasındaki g(x,y) ve f(x,y) fonksiyonlarının grilik seviyesidir.



Şekil 4.1 Görüntü üzerinde bulunan herhangi bir (x,y) noktasının 3 x 3'lük komşuluğu.



Şekil 4.2 Kontrast Zenginleştirilmesi için grilik seviyesi transformasyon fonksiyonu.

Bu transformasyonlar uygulandığı takdirde, görüntü maskeleme yapılmış olur. Genelde, maskeleme (örneğin 3x3'lük komşuluk gibi) iki boyutlu dizinlerde yapılmaktadır. Bu kavram sabit ışık şiddeti üzerine değişik ışık şiddeti değerlerinin atanması ile elde edilir(1,4,5).

-1	-1	-1
-1	8	-1
-1	-1	-1

Şekil 4.3 Maskeleme katsayıları.

Şekil 4.3'de gösterildiği gibi maskeleme katsayıları kullanılarak, tüm görüntü üzerinde bulunan noktalar pixel-pixel çarpılarak, maskeleme elde edilmiş olur(1,4).

w1 (x-1,y-1)	w2 (x-1,y)	w3 (x-1,y+1)
w4 (x,y-1)	w5 (x,y)	w6 (x,y+1)
w7 (x+1,y-1)	w8 (x+1,y)	w9 (x+1,y+1)

Şekil 4.4 Genel 3 x 3 'lük maskeleme katsayıları ve pixellerin yerleşimi.

Sekiz komşuluğu bulunan ve merkezi (x,y) olan bir noktanın maskelenmesi, Şekil 4.4'de gösterilmiştir.

$$T[f(x,y)] = w1*f(x-1,y-1) + w2*f(x-1,y) + w3*f(x-1,y+1) + w4*f(x,y-1) + w5*f(x,y) + w6*f(x,y+1) + w7*f(x+1,y-1) + w8*f(x+1,y) + w9*f(x+1,y+1)$$

ifadesi ile meydana getirilir. Daha büyük boyutlarda (5 x 5, 7 x 7 veya 9 x 9 gibi) maskeleme işlemleri, yukarıdaki şekilde formüle edilebilir. Maskeleme katsayıları değiştirilerek görüntü üzerinde istenilen değişiklik yapılabilir.

4.1.2 Frekans Alanı Yöntemleri

Frekans alanı metodları konvolüsyon teoremi yardımıyla ifade edilir:

$$g(x,y) = h(x,y) * f(x,y)$$

Burada g(x,y) transformasyonun sonucu, f(x,y) konvolüsyon uygulanacak orijinal görüntü ifadesi ve h(x,y) değişmeyen pozisyon ifadesidir. Yukarıda belirtilen ifade frekans alanında tanımlanırsa:

$$G(u,v) = H(u,v)F(u,v)$$ olur.

Burada G,H,F fonksiyonları sırası ile g,h,f'nin Fourier transformlarıdır. H(u,v) transformasyonu işlemin transfer fonksiyonu olarak adlandırılır. Görüntü zenginleştirilmesi uygulamalarında verilen f(x,y) giriş fonksiyonu frekans alanına çevrildikten ve H(u,v) seçildikten sonra istenen görüntü fonksiyonu (ters fourier transformu FT^{-1} ile gösterilmiştir.)

$$g(x,y) = FT^{-1}[H(u,v)*F(u,v)]$$ olur.

Uzaysal alan yöntemlerinde kullanılan matematiksel ifadeler, frekans alanında FFT (Fast Fourier Transform) kullanılırsa,

islem sayısı azalır ve islem hızı da artar. Discrete Fourier Transformasyonu gereğince H(u,v) ve h(x,y)'nin esit boyutlarda olması gerekmektedir. Bu da FFT yardımıyla frekans alanında yapılan islemlerde görüntü üzerinde bazı bozulmalar meydana getirir. Bundan dolayı sık sık interpolasyon tekniğine baş vurulmustur.

4.2 Histogram Yardımı ile Görüntü Zenginlestirilmesi

Histogram, ekran üzerinde görünen görüntünün grilik seviyesini gösterir.

4.2.1 Temel Yapı

Zenginlestirilmis görüntü üzerindeki bir pixelin grilik seviyesinin degisken r ile ifade edildiği varsayılırsa r için

$0 \leqslant r \leqslant 1$ esitliği geçerli olur.

ve gri ölçek üzerinde $r = 0$ iken siyah, $r = 1$ iken beyazın ve $[0,1]$ aralığında herhangi bir r değeri için

$s = T(r)$ yazılır.

Burada s, orijinal görüntü üzerinde bulunan her bir pixel değeri r için oluşan seviyedir. s transformasyon fonksiyonu Şekil 4.5'de gösterildiği gibi aşağıdaki şartları sağlamalıdır:

a) T(r) tek değerli olmalı ve $0 \leqslant r \leqslant 1$ aralığında artmalıdır.

b) $0 \leqslant r \leqslant 1$ için $0 \leqslant T(r) \leqslant 1$ olmalıdır.

Şart a) için gri ölçek üzerinde siyahdan beyaza doğru olusan ölçeğin düzeni ve Şart b) için pixel değerlerinin istenilen bölge içinde bulunması gerekmektedir.

s'nin ters transformunu alınırsa

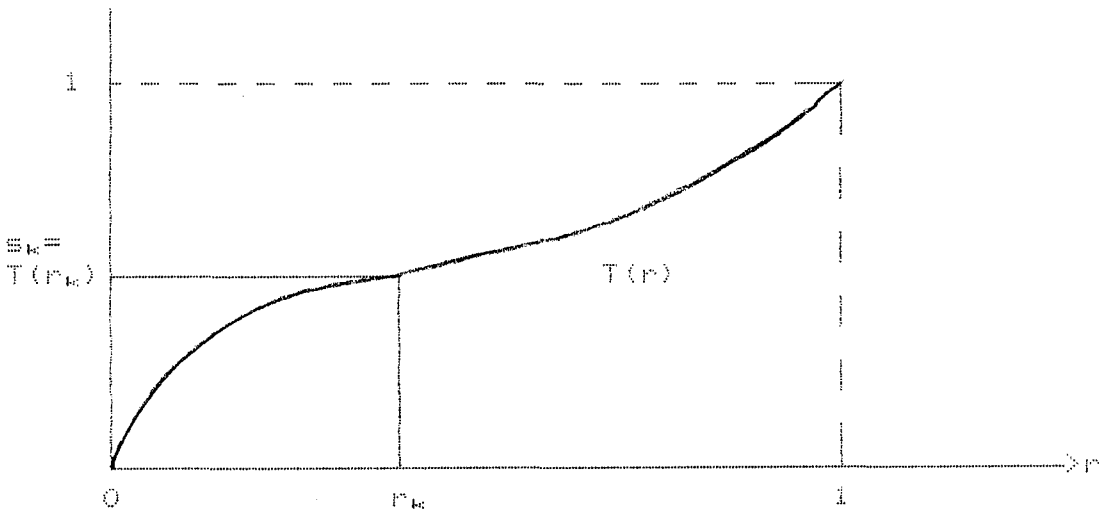
$$r = T^{-1}(s) \qquad 0 \leq s \leq 1$$

olur. Yukarıda belirtilen şartlar T için de geçerlidir.

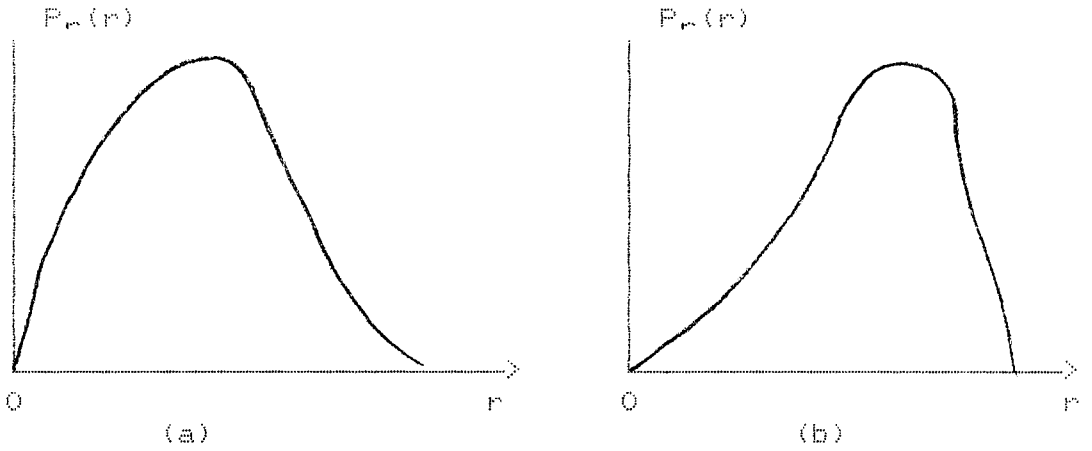
Görüntü üzerindeki grilik seviye dağılımı [0,1] aralığında rastgele olarak dağılmaktadır. Böylece Şekil 4.6'da görüldüğü gibi grilik seviyeleri olasılık yoğunluk fonksiyonu $P_r(r)$ ve $P_s(s)$ ile karakterize edilir. Bu olasılık fonksiyonları ve transformasyon arasındaki ilişki,

$$P_s(s) = \left[P_r(r) \frac{dr}{ds} \right]_{r = T^{-1}(s)}$$

ile gösterilir.



Şekil 4.5 Grilik seviyesi transformasyon fonksiyonu.



Şekil 4.6 Grilik seviyesi olasılık yoğunluk fonksiyonu

(a) karanlık (b) aydınlık görüntü

4.2.2 Histogram Eşitliği

Transformasyon fonksiyonunu şöyle tanımlanabilir:

$$s = T(r) = \int_0^r P_r(w)dw \qquad 0 < r < 1$$

burada w, integrasyonun dummy değişkenidir. Yukarıdaki eşitliğin sağ tarafı (integrasyon kısmı) r'nin Kumulatif Dağılım Fonksiyonu (KDF)'dur. Bölüm 4.2.1'de anlatılan şartlar KDF için de geçerli olup aşağıdaki şekilde tanımlanır:

$$\frac{ds}{dr} = P_r(r) \quad \text{ve}$$

$$P_s(s) = \left[P_r(r) \frac{1}{P_r(r)} \right]_{r = T^{-1}(s)}$$

$$= [\ 1\ ]_{r = T^{-1}(s)}$$

$$= 1 \qquad 0 < s < 1$$

olur.

Örnek: $$P_r(r) = \begin{cases} -2r + 2 & , 0 < r < 1 \\ 0 & , \text{diğer yerlerde} \end{cases}$$

$$s = T(r) = \int_0^r (-2w + 2)dw$$

$$= -r^2 + 2r$$

çıkar buna rağmen bize histogram eşitliği için yalnızca T(r) lazımdır:

$$r = T(s) = 1 \pm \sqrt{1 - s}$$

[0,1] aralığı için çözüm,

$$r = T(s) = 1 - \sqrt{1 - s} \quad \text{olur.}$$

s'nin olasılık yoğunluk fonksiyonu,

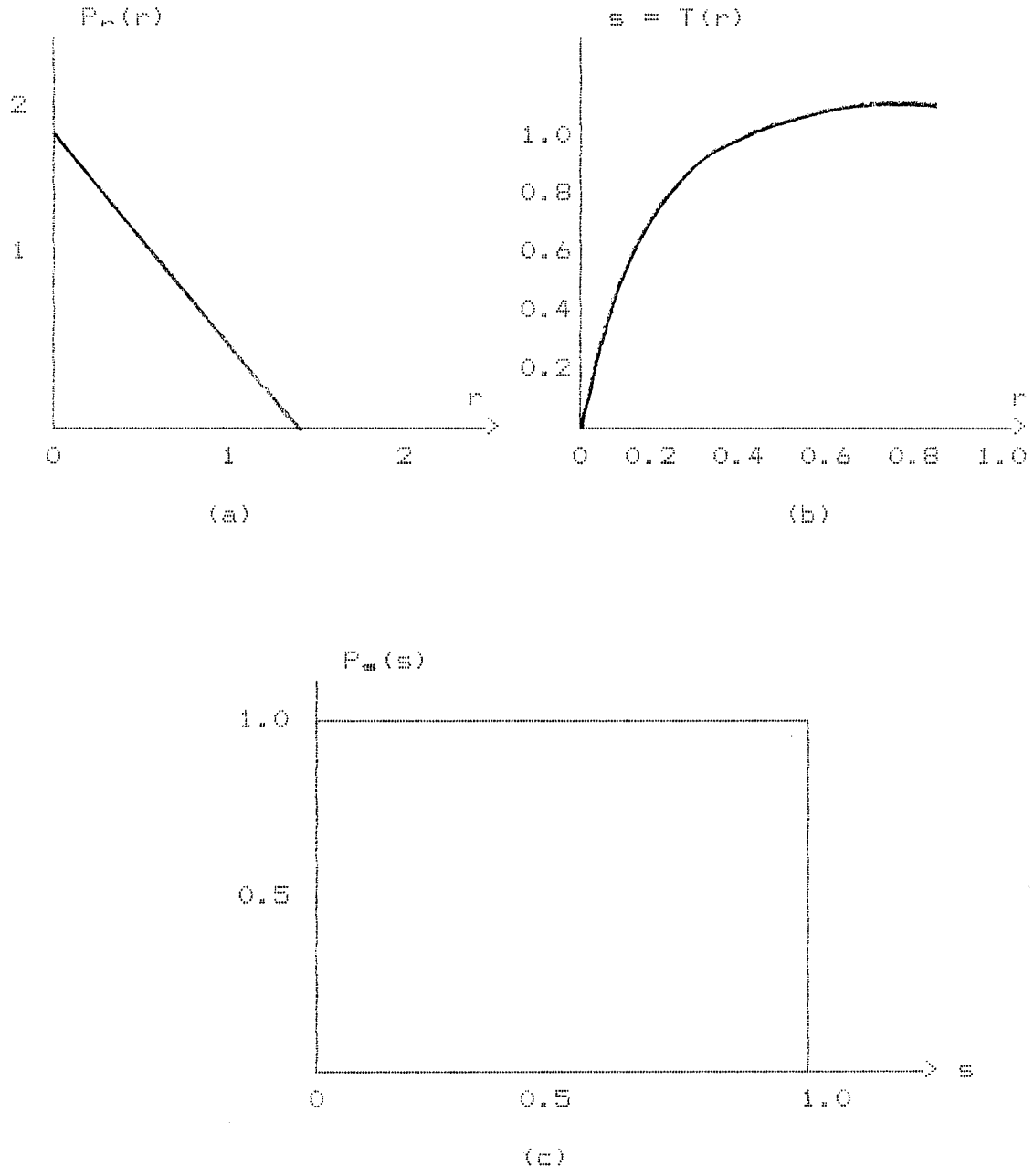
$$P_s(s) = \left[P_r(r) \frac{dr}{ds} \right]_{r = T^{-1}(s)}$$

$$= \left[(-2r + 2) \frac{dr}{ds} \right]_{r = 1 - \sqrt{1 - s}}$$

$$= \left[(2\sqrt{1 - s}) \frac{d}{ds} (1 - \sqrt{1 - s}) \right]$$

$$= 1 \qquad 0 < s < 1$$

olur. Böylece düzgün yoğunluk istenilen bölgededir. Transformasyon fonksiyonu T(r) Şekil 4.7 (b) ve $P_s(s)$ Şekil 4.7 (c) 'de görülmektedir.



Şekil 4.7 Düzgün yoğunluk transformasyon metodu.

a) orijinal olasılık yoğunluk fonksiyonu.

b) Transformasyon fonksiyonu.

c) sonuç düzgün yoğunluk.

Grilik seviyeleri için oluşan olasılık fonksiyonu aşağıdaki şekilde tanımlanır:

$$P_r(r_k) = \frac{n_k}{n} \qquad \begin{array}{l} 0 < r_k < 1 \\ k = 0,1,\ldots,L-1 \end{array}$$

Burada L grilik seviyesi, $P_r(r_k)$ k'ıncı grilik seviyesinin olasılığı, n_k görüntü üzerinde görülen grilik seviyesinin zamanı ve n görüntü üzerinde bulunan toplam pixel sayısıdır. $P_r(r_k)$'nın r_k'ya göre çizimine histogram denir ve düzgün histogram teknikleri kullanıldığında bilinen histogram eşitliği veya histogram lineerleştirilmesi ortaya çıkar.

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n}$$

$$= \sum_{j=0}^{k} P_r(r_j) \qquad \begin{array}{l} 0 < r_k < 1 \\ k = 0,1,\ldots,L-1 \end{array}$$

ve ters transformasyonu,

$$r_k = T^{-1}(s_k) \qquad 0 \leq s_k \leq 1$$

olur. $T(r_k)$ ve $T(s_k)$, Bölüm 4.2.1'de anlatılan şartları sağlar.

Örnek: 64 x 64 ve sekiz seviyeli bir görüntünün grilik seviye dağılımı Tablo 4.1'de verilmiştir. Bu grilik seviyesinin histogramı Şekil 4.8'de görülmektedir.

Tablo 4.1

r_k	n_k	$P_r(n_k) = n_k/n$
r0 = 0	790	0.19
r1 = 1/7	1023	0.25
r2 = 2/7	850	0.21
r3 = 3/7	656	0.16
r4 = 4/7	329	0.08
r5 = 5/7	245	0.06
r6 = 6/7	122	0.03
r7 = 1	81	0.02

Çözüme ilk değerden başlanırsa:

$$S0 = T(r0) = \sum_{j=0}^{0} P_r(r_j)$$

$$= P_r(r0)$$

$$= 0.19$$

Benzer şekilde,

$$S1 = T(r1) = \sum_{j=0}^{1} P_r(r_j)$$

$$= P_r(r0) + P_r(r1)$$

$$= 0.44 \quad \text{olur.}$$

ve

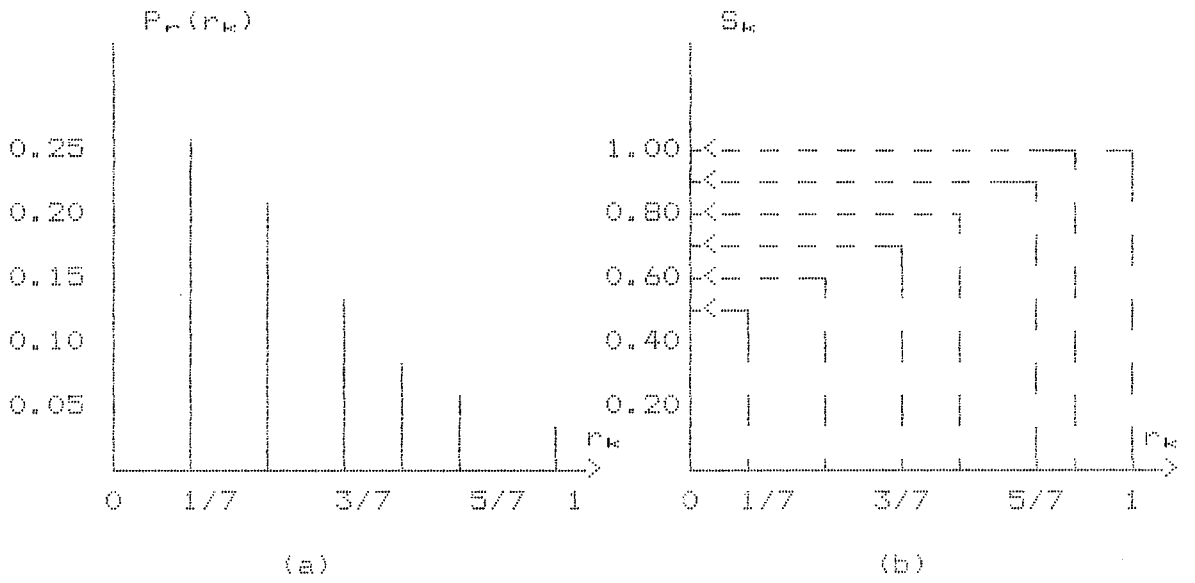
S2 = 0.65 S5 = 0.95

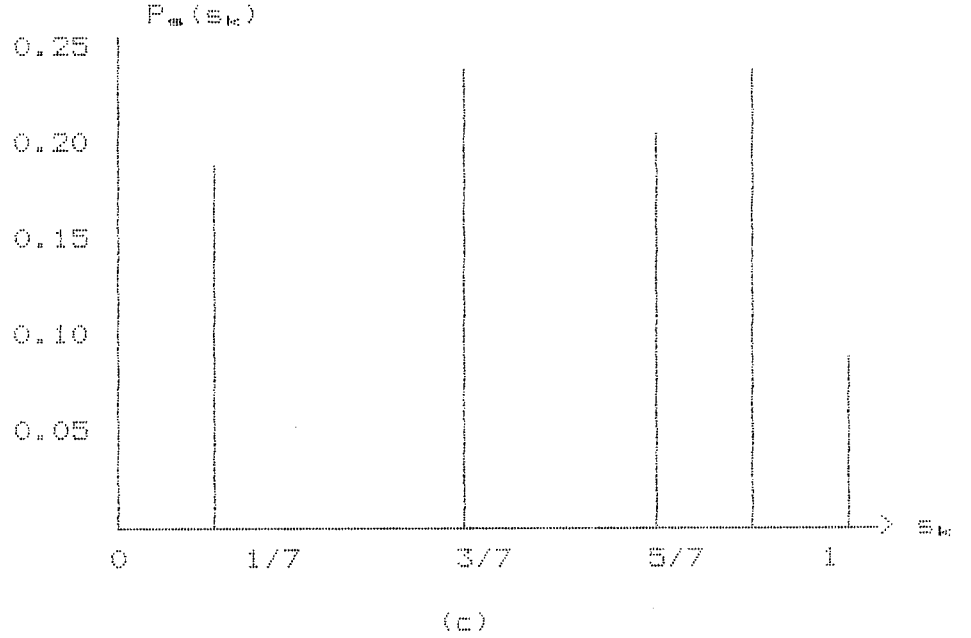
S3 = 0.81 S6 = 0.98

S4 = 0.89 S7 = 1

Transformasyon fonksiyonu Şekil 4.8 (b) 'de görülmektedir.

Her bir dönüşüm değeri kapalı bir seviyede geçerli olarak





Şekil 4.8 Histogram eşitliği

a) Orijinal histogram

b) Transformasyon fonksiyonu

c) Eşitlenmiş histogram

atanırsa,

S0 = 1/7 S4 = 6/7

S1 = 3/7 S5 = 1

S2 = 5/7 S6 = 1

S3 = 6/7 S7 = 1 olur.

Böylece Sekil 4.8 (c)'de görüldüğü gibi yalnızca beş histogram eşitliği grilik seviyesi oluşur ve bu aşağıdaki değerlere karşılık gelmektedir.

S0 = 1/7

S1 = 3/7

S2 = 5/7

S4 = 6/7

S5 = 1

4.3 Görüntü Düzgünleştirilmesi

4.3.1 Komşuluk Ortalaması

Komşuluk ortalaması, görüntü düzgünleştirilmesi için bir uzaysal alan tekniğidir. Verilen N x N boyutuyla bir f(x,y) görüntüsünün her bir (x,y) noktasının grilik seviyesinin düzgünleştirilmesi ile bir g(x,y) fonksiyonu oluşmaktadır.

$$g(x,y) = \frac{1}{M} \sum_{(n,m)\in S} f(n,m) \quad , \; x,y = 0,1,\ldots,N-1$$

Burada S_{xy} (x,y) noktasının komşuluğundaki noktaların koordinatlarını içeren küme ve M komşuluk içinde bulunan toplam nokta sayısıdır. Eğer özel olarak seçilmiş bir "threshold" değeri alınırsa, fonksiyon

$$g(x,y) = \begin{cases} \dfrac{1}{M} \sum_{(m,n)\in S} f(m,n) & \text{, eğer } \left| f(x,y) - \dfrac{1}{M} \sum_{(m,n)\in S} f(m,n) \right| < T \text{ ise} \\ f(x,y) & \text{, diğer yerlerde} \end{cases}$$

olur.

Burada T negatif olmayan özel "threshold" değeridir. Bu çalışmada görüntüler üzerinde yapılan denemeler sonucunda T=10 en uygun değer kabul edilmiştir.

4.3.2 Orta Değer Filitrelemesi

Pixelin komşuluğunda bulunan diğer pixellerin grilik seviyesinin orta değerinin alınması ile elde edilen filitrelemedir. Komşuluk sayısı artması ile görüntü üzerindeki yumusama daha fazla olur. Teknik olarak, merkezi (x,y) olan bir noktanın komşuluğunda bulunan diğer nokta değerlerinin, büyükten küçüğe veya kücükten büyüğe doğru sıralanması ile olusmakta ve orta değer o noktanın yeni değerini vermektedir.

4.3.3 Alçak Geçiren Filtre

Grilik seviyesi üzerinde görülen gürültü miktarını gidermek amacıyla Fourier transformu kullanılarak yüksek frakansların giderildiği bir yöntemdir. Çıkış fonksiyonun transformu, G(u,v):

$$G(u,v) = H(u,v)F(u,v)$$

olarak verilmistir. Burada F(u,v) giriş fonksiyonunu transformu, H(u,v) özel olarak seçilmiş ve yüksek frekansları yok edebilen transformasyon fonksiyonudur. G(u,v)'nin ters fourier transformu alındığında, yüksek frekanslardan ayrılmış ve sadece düşük frekanslar içeren g(x,y) fonksiyonu elde edilir ve

$$g(x,y) = FT^{-1}\,[H(u,v)F(u,v)]$$

olarak yazılır.

Ideal Filtre

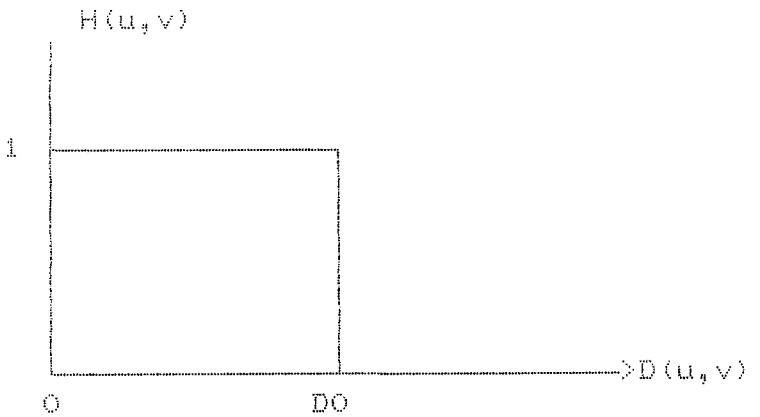
İki boyutlu ideal alçak geçiren filtrenin (ILPF) transfer fonksiyonu:

$$H(u,v) = \begin{cases} 1 & ,\text{eğer } D(u,v) < DO \text{ ise} \\ 0 & ,\text{eğer } D(u,v) > DO \text{ ise} \end{cases}$$

şeklinde yazılabilir. Burada DO negatif olmayan özel bir değer, D(u,v)'de (u,v) noktasından frekans düzleminin orijinine olan uzaklık fonksiyonudur. Bu fonksiyon şekil 4.9'da gösterilmiş olup

$$D(u,v) = (u^2 + v^2)^{1/2}$$

olarak bulunur.



Şekil 4.9 Kesim Frekansı. DO kesim Frekans değeri.

Bu transfer fonksiyonunun 1 ile 0 arasındaki geçiş noktasına kesim Frekansı denir.

BÖLÜM 5

PROGRAM ve ÖZELLIKLERI

Bu program, daha önceden görüntünün digitizer'dan alınmış dijital değerlerini ekranda işleme görevini yapmaktadır. Program, biri ana program, üçü yardımcı program olmak üzere dört kısımdan oluşmuştur. Bu dört program tamamen birbirinden bağımsız olarak çalışmaktadır.

Bu projede Tomografi ve Ultrasonografi cihazlarında bulunan görüntüleme özellikleri göz önüne alınarak programlar yazılmıştır. Bu cihazların görüntüleme kısımları aynı olup yalnızca görüntünün oluşması için gereken bilgi değişik yöntemlerle elde edilmiştir. Örneğin Tomografi cihazı X-ışını kullanırken Ultrasound cihazı ses dalgalarını kullanmaktadır(2,4,5). Meydana getirilen programın en iyi şekilde çalışması için gerekli olan donanım ve yazılım konfigurasyonu aşağıdaki özelliklere sahip olmalıdır:

- IBM uyumlu PC/XT/AT bilgisayar,
- VGA ve MCGA grafik kartının olması,
- Ana hafızanın en az 640KB olması,
- DOS işletim sisteminin olması,
- ROM BIOS programlarının tamamının bulunması(128KB ROM)
- Renkli monitora sahip olması

Bu şartları taşımayan bir bilgisayar kullanıldığı takdirde program çıktılarının istenilen özellikleri mümkün olmamaktadır.

5.1. Ana Program

Bu program, digitizer'dan alınmış verileri, IMG dosya

uzatması ile disket üzerinde bulur ve diğer üç yardımcı programın ön sartlarını hazırlar. Bu ana program çalışma prensibi olarak:

- Yardımcı programlardan hangisinin kullanılacağını,
- Hangi görüntünün seçileceğini ve
- Yardımcı programlarda kullanılan parametrelerin ne şekilde set edileceğini sağlamaktadır.

5.2 Yardımcı Programlar

5.2.1 MCGA ve VGA'de Çalışabilen Program

Bu yardımcı program 64 gri ölçek kullanılarak yazılmıştır. Ekran boyutları 320 x 200 ve ekranda oluşan görüntünün boyutları ise 250 x 200 pixelden oluşmaktadır. Bu programda düzgünleştirme tekniklerinden orta değer filtresi ve komşuluk ortalaması filitresi kullanılmıştır. Bu filtrelerde pixelin değişik komşulukları seçilmiştir(3 x 3, 5 x 5, 7 x 7 ve 9 x 9'lu komşuklar). Ayrıca pencere ayarları olan pencere genişliği ve pencere seviyesi konulmuştur. Böylece kullanıcı görüntüyü istediği şekilde görebilme imkanına sahip olmaktadır.

5.2.2 Yalnızca VGA'da Çalışan Program

Bu programda, dijital görüntüleme sistemlerinde kullanılan aşağıda verilen tüm özellikleri içerecek şekilde yazılmıştır. Programda ekranın en üst satırı menü ve en alt satırı ise yardımcı menü satırı olarak seçilmiştir.

Enlargement : Ekran üzerinde bulunan görüntünün istenilen bölgesinin büyütülmesi için oluşturulan bir fonksiyondur. Bu bölüm iki alt gruba ayrılmaktadır.

- X2 Enlarge: İstenilen bölgenin dört kat büyütülmesi.

- X4 Enlarge: İstenilen bölgenin sekiz kat büyütülmesi.

Transposition : Ekran üzerinde bulunan görüntünün yatay ve dikey olarak yer değiştirmesini sağlayan bir fonksiyondur. Bu bölüm iki alt gruba ayrılmaktadır:

- UP/DOWN: Görüntünün yatay olarak (Vertical Mirror) döndürülmesidir.

- LEFT/RIGHT: Görüntünün dikey olarak (Horizantal Mirror) döndürülmesidir.

Histogram : Ekran üzerinde bulunan görüntünün istenilen bölgesinin grilik seviyesinin fonksiyonunu cizmektedir. Bu bölüm üç alt gruba ayrılmaktadır:

- X2:Görüntünün dörte biri kadar olan herhangi bir bölgenin histogramının çizilmesi,

- X4:Görüntünün sekizde biri kadar olan herhangi bir bölgenin histogramının çizilmesi ve

- ALL:Görüntünün tamamının histogramını çizilmesidir.

Filter : Ekran üzerinde bulunan görüntünün istenilen filitreden geçirilmesi islemidir. Bu bölüm de dört alt gruba ayrılmaktadır:

- FC1: 3 x 3 komşuluk ortalaması,

- FC2: 5 x 5 komşuluk ortalaması,

- FC3: 3 x 3 orta değer filitresi ve

- FC4: 5 x 5 orta değer filitresidir.

Band Filter : Ekran üzerinde bulunan görüntünün renklendirilmesi için olusturulan fonksiyondur. Renklendirme kullanıcının isteğine bırakılmıştır.

5.2.3 Özel Filtreler

Bu programda görüntü düzgünlestirilmesi ve görüntü zenginlestirilmesi yöntemleri kullanılmıstır. Tüm filtrelerde 3 x 3 komsuluk esas alınmıstır. Bu bölüm altı alt gruptan olusmaktadır.

- F1 : Original görüntüyü ekranda görüntüleme,
- F2 : Heavy smoothing filitreleme,
- F3 : Smoothing filitreleme,
- F4 : Light edge enhancement filitreleme,
- F5 : Edge enhancement filitreleme ve
- F6 : Heavy edge enhancement filitreleme.

Bu filtrelerde kullanılan maskeleme katsayıları arastırmalar ve denemeler sonucunda elde edilmistir(3,4,5). Bu katsayılar asagıda verilen tablolarda gösterilmistir:

- HEAVY SMOOTHING:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- SMOOTHING:

0	1/8	0
1/8	3/8	1/8
0	1/8	0

- LIGHT EDGE ENHANCEMENT:

0	-1	0
-1	9/2	-1
0	-1	0

- EDGE ENHANCEMENT:

0	-2	0
-2	17/2	-2
0	-2	0

- HEAVY EDGE ENHANCEMENT:

0	-2	0
-2	9	-2
0	-2	0

Bu katsayıların değiştirilmesi ile görüntü üzerinde istenilen değişiklik yapılabilmektedir. Bunun için programda kullanıcının istediği değerleri girmesi için ayrı bir bölüm oluşturulmuştur. Bu bölümün formatı Şekil 5.1'de gösterilmiştir. Burada bulunan katsayılar tamamen kullanıcının isteğine bağlıdır. Bu bölümü kullanabilmek için önceden istenilen katsayıların girilmesi gerekmektedir.

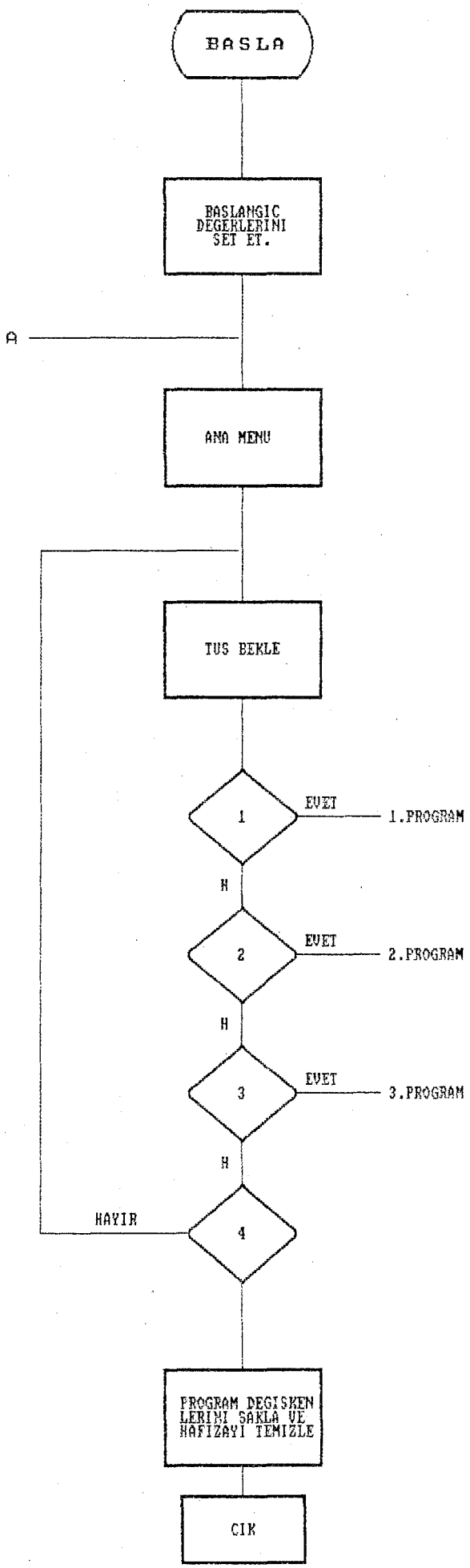
B	C	B
C	A	C
B	C	B

Şekil 5.1 Kullanıcı maskeleme katsayıları.

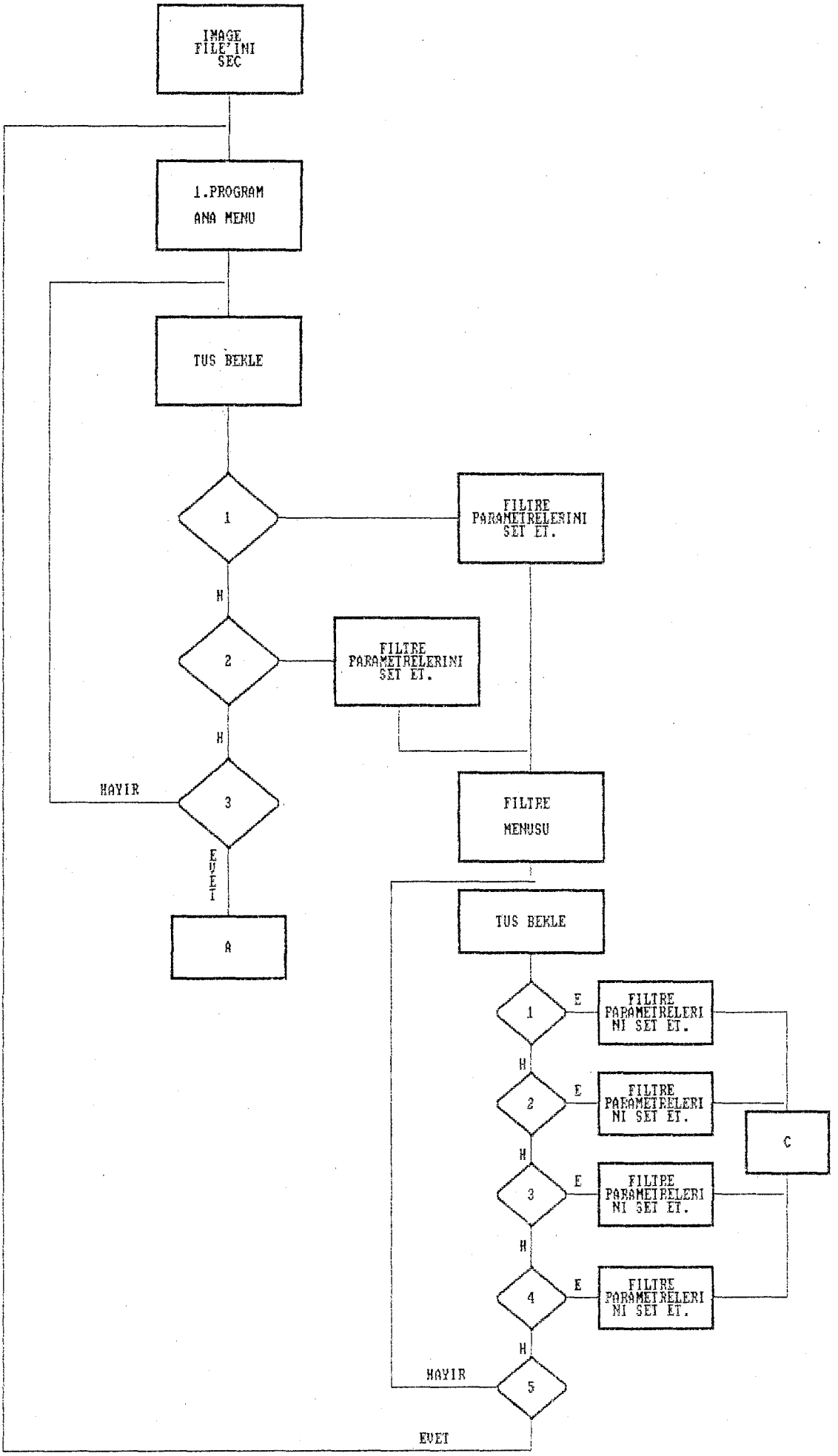
5.3. Yazılım ve Akış Diyagramları

Meydana getirilen program yüksek seviyeli programlama dili olarak bilinen Turbo Pascal ve düşük seviyeli programlama dili olarak bilinen Assembler (Intel 8086) dilleri kullanılarak oluşturulmuştur. Programın tamamı modüler olarak tasarlanmış ve hafızada fazla yer tutmaması için bölümlere ayrılmıştır. Ayrıca virüslenmeye karşı koruma altına alınmıştır.

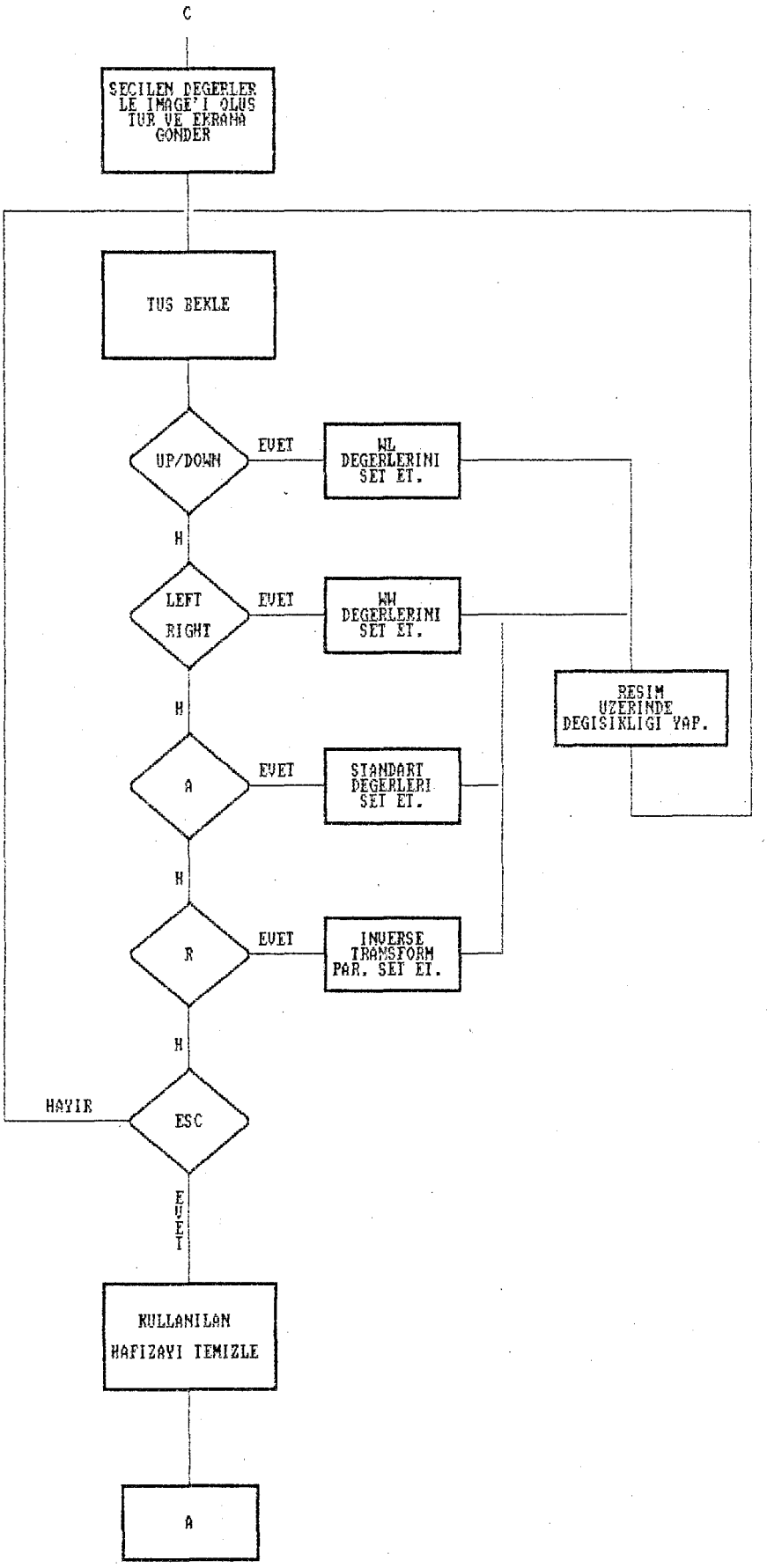
Ana program Şekil 5.2'de gösterildiği gibi başlangıçda kendisinin ve yardımcı programların çalışması için gerekli olan hafıza bölgesi seçilmekte ve ön şartlar sağlanmaktadır. Kullanıcı, istediği yardımcı programı ve görüntüyü seçmesi için ana menüyü ekrana getirmektedir. Tüm işlemler bu ana program üzerinden yapılmaktadır. Seçilen yardımcı programlar, Şekil 5.3, 5.4 ve 5.5'de gösterildiği gibi tanımlanan özellikler doğrultusunda çalışmakta olup, işlemleri bittiğinde kullandıkları hafıza alanlarını ve diğer program parametrelerini temizleyerek ana programa geri dönerler.



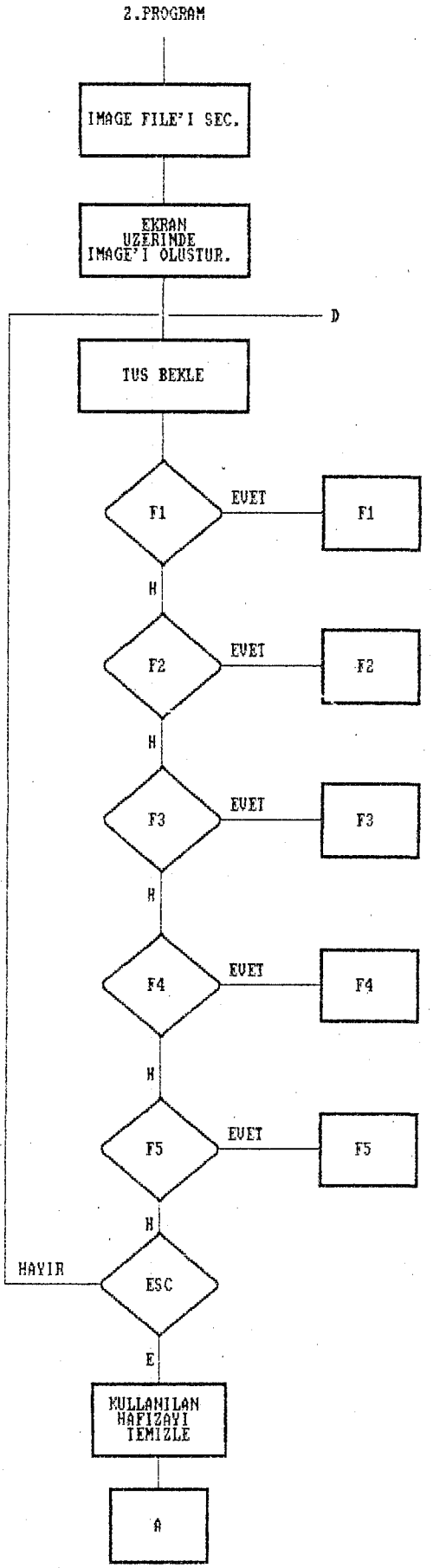
Sekil 5.2 Ana program akis diyagrami



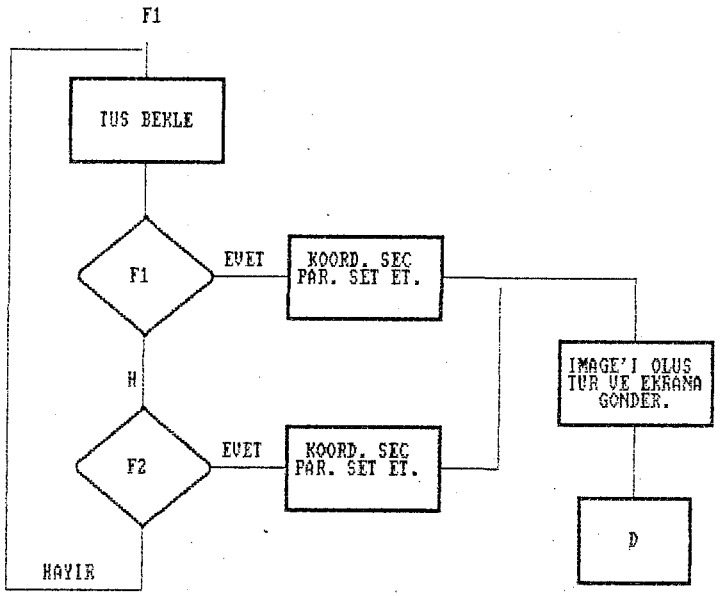
Sekil 5.3 1.Program akış diyagramı

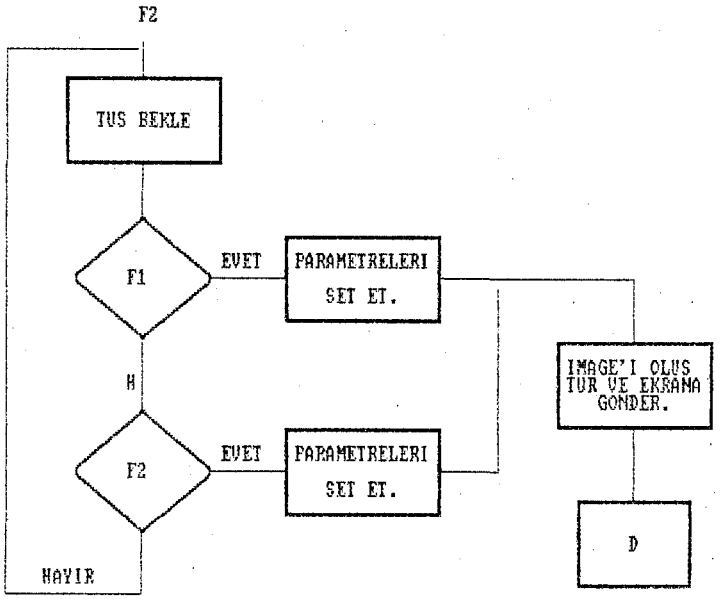


Şekil 5.3 (Devam)

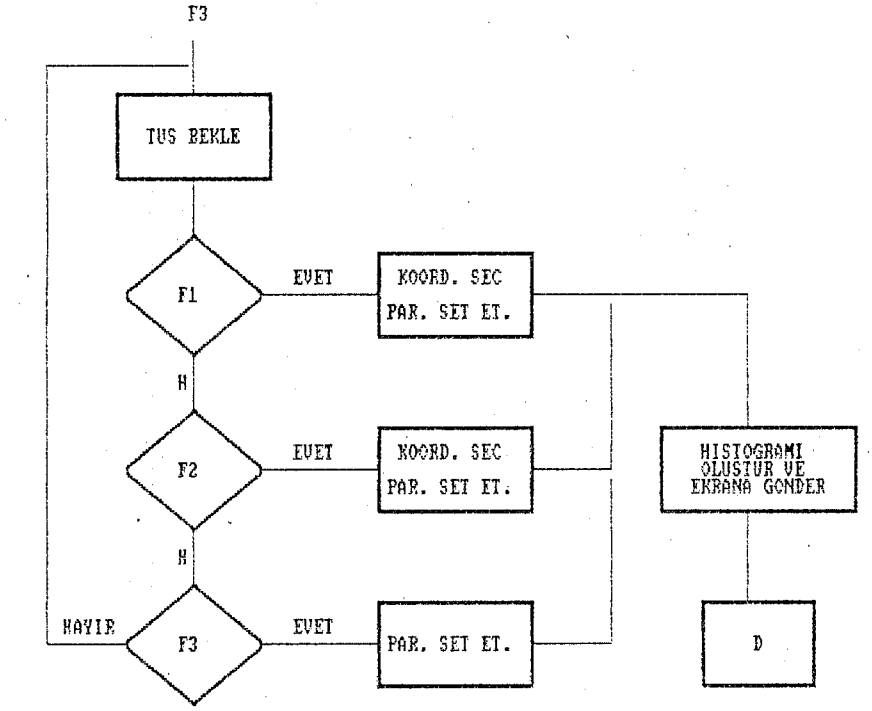


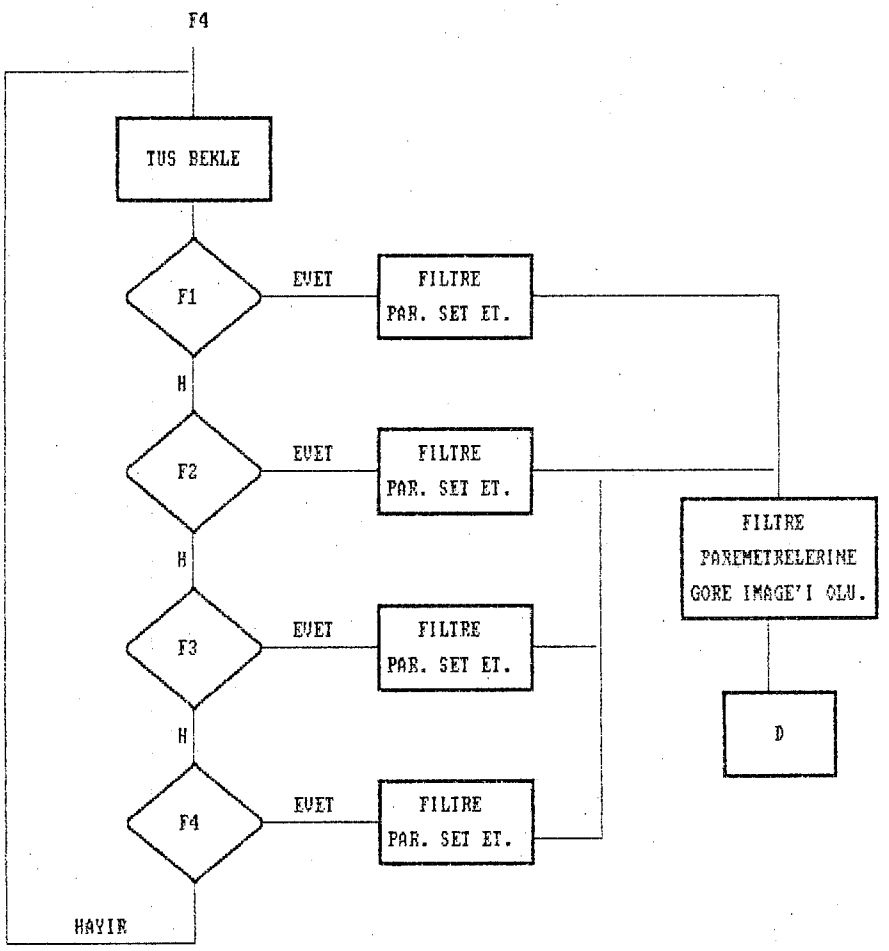
Şekil 5.4 2.Program akış diyagramı



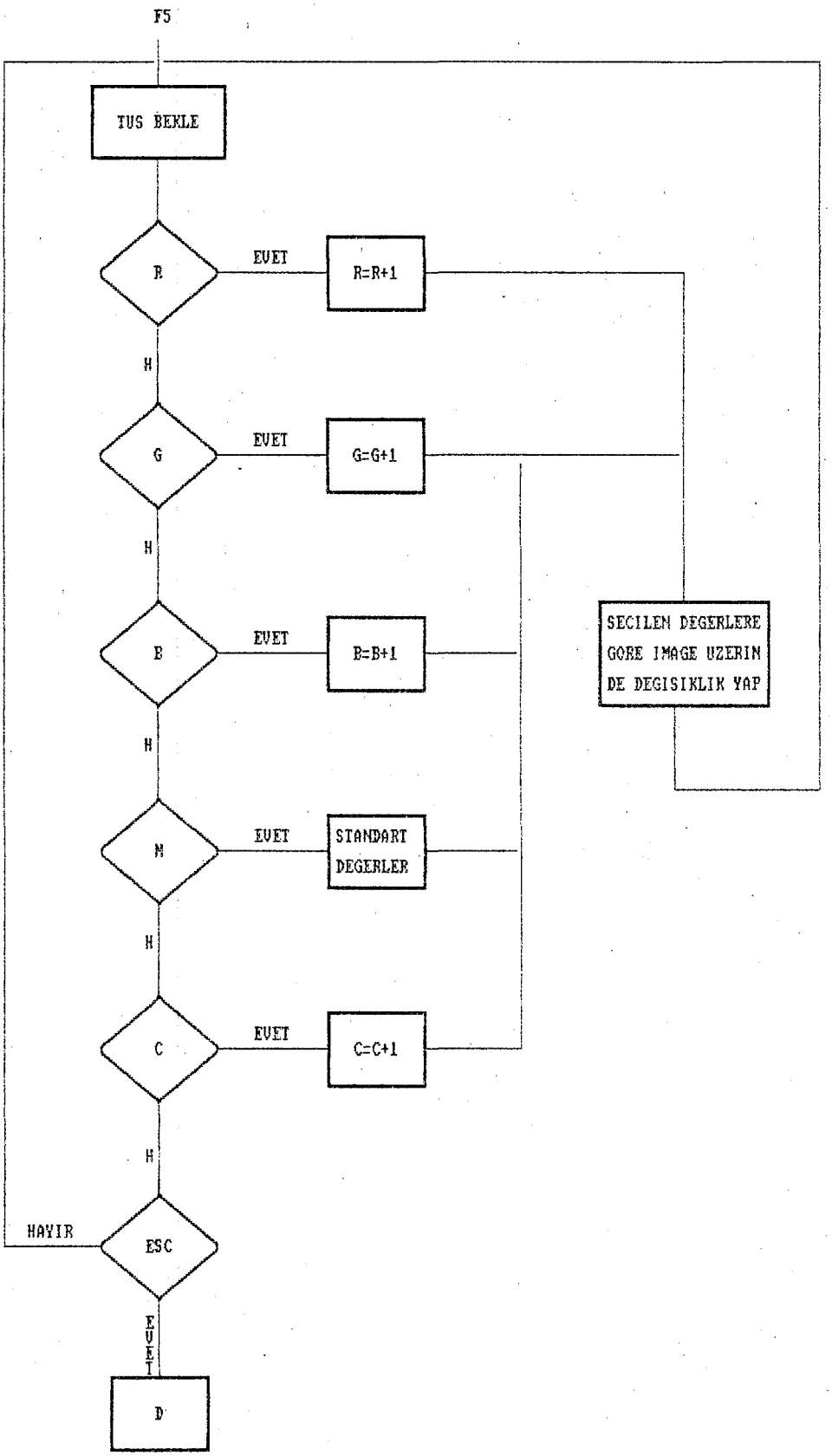


Sekil 5.4 (Devam)

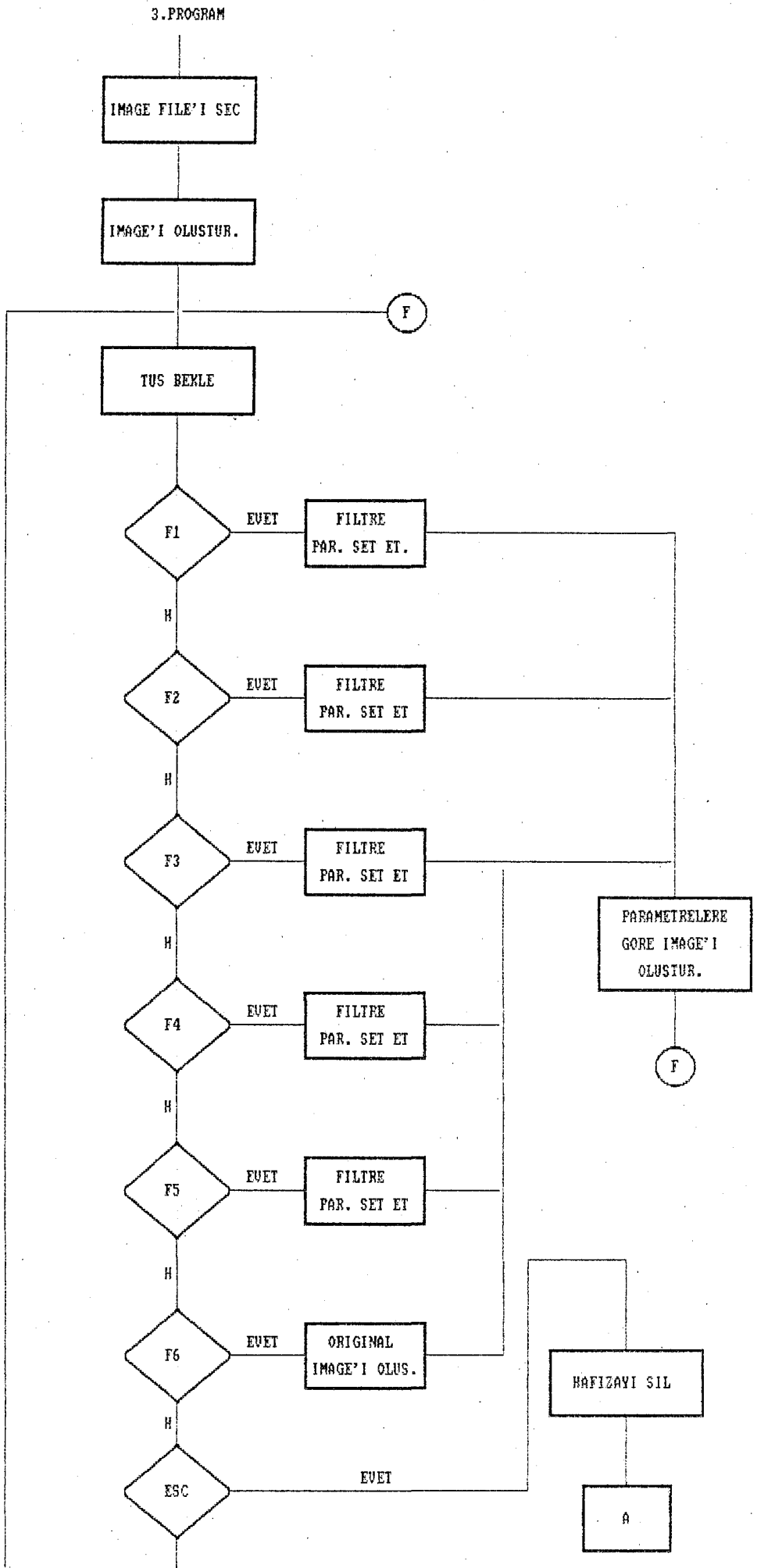




Sekil 5.4 (Devam)



Sekil 5.4 (Devam)



Şekil 5.5 3. Program akış diyagramı

BÖLÜM 6

SONUÇ ve ÖNERİLER

Bu projede gerçekleştirilmek istenen amaç, IBM PC'yi tomografi cihazına bağlamak ve bu cihaza ait görüntüleme işlemlerini IBM PC yardımı ile yapmaktan ibarettir. Bu olayı gerçekleştirmek için yukarıdaki cihazların birbirleriyle haberleşmesini sağlamak gerekmektedir. Bu haberleşmeyi sağlamak için de iki cihazı modem aracılığı ile birbirine bağlamak lazımdır. Bundan dolayı her iki cihazın yazılım ve donanım olarak çok iyi bilinmesi gereklidir. Bu konuda Toshiba marka bilgisayarlı tomografi cihazına ait tüm dökümanlar ve elektriksel diyagramlar incelenmiştir. Bu kaynaklarda yeterli bilgi verilmediğinden dolayı, direkt olarak Toshiba genel merkezinden bilgi istenilmesine rağmen olumlu bir sonuç alınamamıştır.

Bu projede cihazlar arası haberleşme sağlanamadığı için istenilen bilgiler simulasyon yöntemi ile elde edilmiştir. Böylece tomografi cihazından alınmış resimler ve normal fotoğraflar digitizer ile dijital resim bilgilerine çevrilmiş ve IBM PC'ye aktarılmıştır. Alınan dijital veriler, yazılan programlarla işlenerek görüntülenmiştir. Bu işlenenen görüntüler ve programlar ekte sunulmuştur.

Bu projede yazılım kısmı olan dijital görüntüleme birimi teknik açıdan istenilen sonucu vermiştir. Bilgisayarlı tomografi cihazına ait bilgiler elde edildiği takdirde küçük bir yazılım değişikliği ile bu projenin asıl amacına ulaşılmış olacaktır.

Bu proje de gelistirilen programlar, bilgisayarlı tomografi haricinde, ultrasonografi, gama kamera, magnetik rezonans ve degisik görüntü tarayıcıları ile kullanılması mümkündür. Kücük bir yazılım degisikligi ile gercek zaman (real time) olarak bilinen hareketli görüntüler islenebilir. Ayrıca, birden fazla ekran bağlantısı yapılarak görüntü ekranlara parçalar halinde dağıtılabilir.

KAYNAKLAR

1. Gonzalez R.C., 1978, Digital Image Processing, London 22-167 p.
2. Toshiba Corp., 1991, CT Change Information. Tokyo.
3. Toshiba Corp., 1991, CT Kurs Notları, Tokyo and İzmir.
4. Toshiba Corp., 1987, CT Service Manuals, Tokyo.
5. Toshiba Corp., 1989, Ultrasound Service Manual's, Tokyo.
6. Toshiba Corp., 1987, Gamma Camera Service Manual's, Tokyo.
7. Toshiba Corp., 1987-1991, CT Electrical Diagrams, Tokyo.
8. Shmadzu Corp., 1986, CT Operating Manual's, Tokyo.
9. Nihon Kohden Corp., 1990, Patient Monitors Service Manual's, Tokyo.
10. Hearn D. and Baker M. P., 1986, Computer Graphics, London 27-77 p.
11. Norton P. and Wilton R., 1988, The IBM PC & PS/2, Washington, 40-67 and 170-184 p.
12. Duncan R., 1988, Advanced MS DOS Programming, Washington, 86-103 and 500-531 p.
13. Göktugan A. H., 1991, IPUP ODTÜ Yüksek Lisans Tezi, Ankara.
14. Borland Corp., 1991, Pascal 6.00 User's and Programmer's Manual, Scotts Valley.
15. Borland Corp., 1991, Turbo C++ User's and Programmer's Manual, Scotts Valley.
16. Graphics Adapters, PC Magazine, June 1991, Volume 10 Number 12, 103-158 p.

17. Graphics Adapters,PC Magazine, July 1991, Volume 10 Number 13, 103-172 p.

18. Olivetti Corp., 1991, PC PRO SX 20 Operation Manual. Italy.

19. Green W.B., 1983, Digital Image Processing-A System Approach, New York.

20. Gonzalez R.C., 1985, Computer Vision, New York, 130-132 p

21. Gerald L. Graef, Graphics Formats, Byte, September 1989, 305-310 p.

22. Phillip Robinson, Variation on a Screen, Byte, July 1991, 251-264 p.

23. Tom Badgett and Corey Sandler, Ultra VGA Debuts on the MikroPaq, Byte, January 1991, 201-206 p.

24. Baldwin L., 1984, Color Consideration, New York, 227-242 p.

25. Bilgisayar Pazarı, Ekim 1991, Sayı 34, 28-53 s.

26. Bilgisayar Pazarı, Ocak 1992, Sayı37, 32-33 s.

27. Bilgisayar Pazarı, Şubat 1992, Sayı 38, 56-60 s.

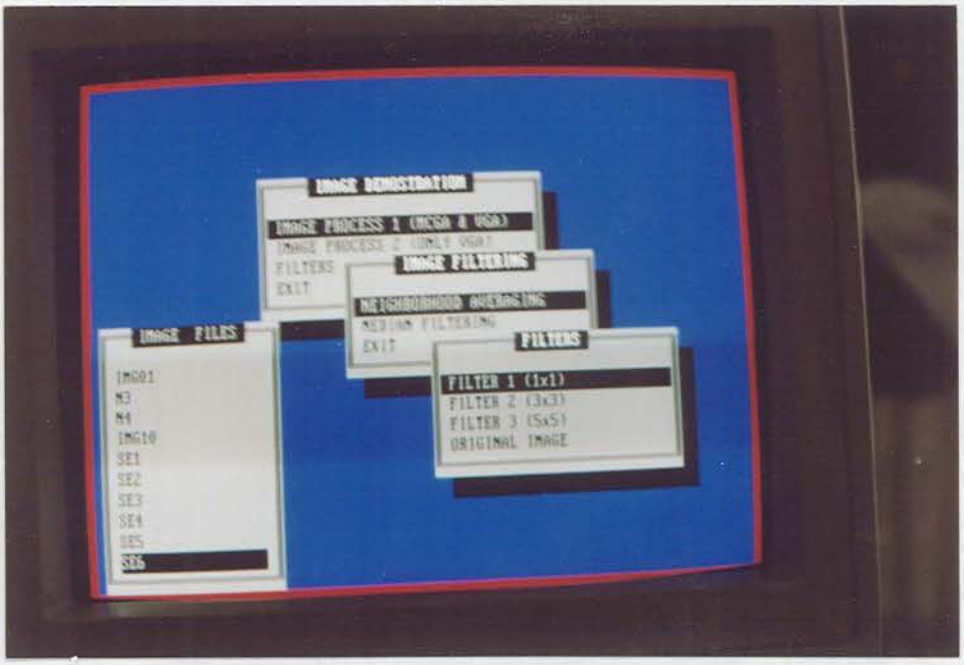
28. Sounds and Images, Byte Dec. 1989, 243-256 p.

29. Jake Richter, XGA a New Graphics Standart, Byte, February 1991, 285-290 p.

30. Radiologic Physics and Pulmonary Radiology, Volume 1, 27,28,29,30,31,32,33,34 chapters.

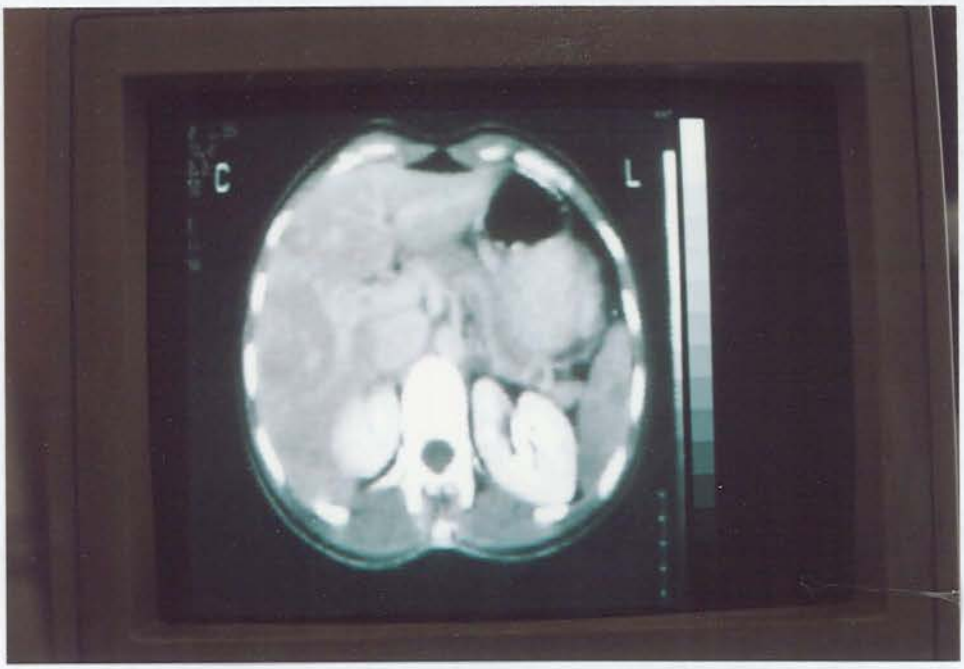
EK 1

PROGRAM ÇIKTILARI

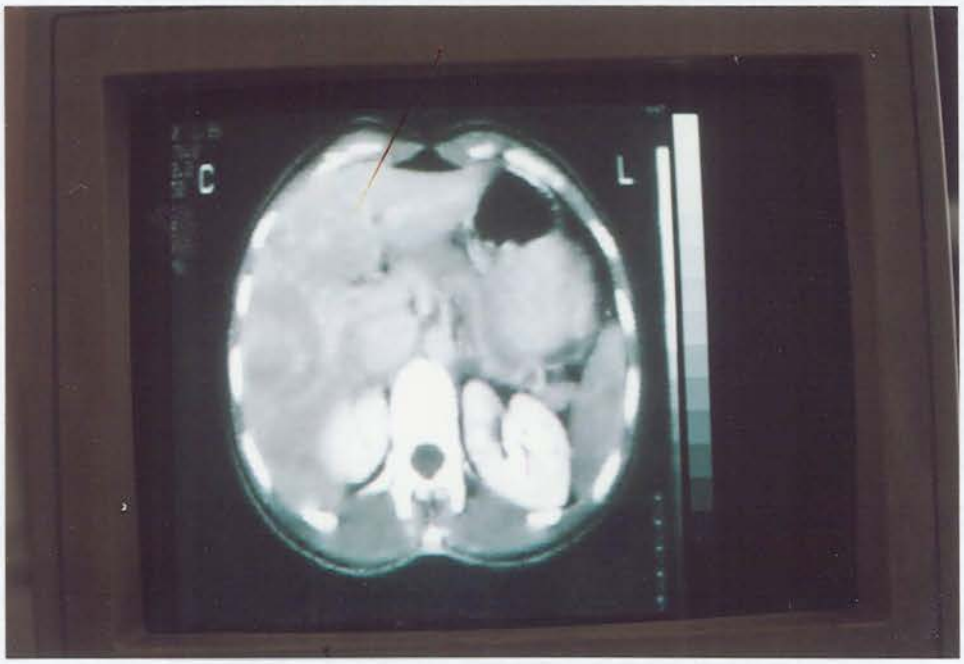


Programın giriş formatı

1. program orijinal görüntü

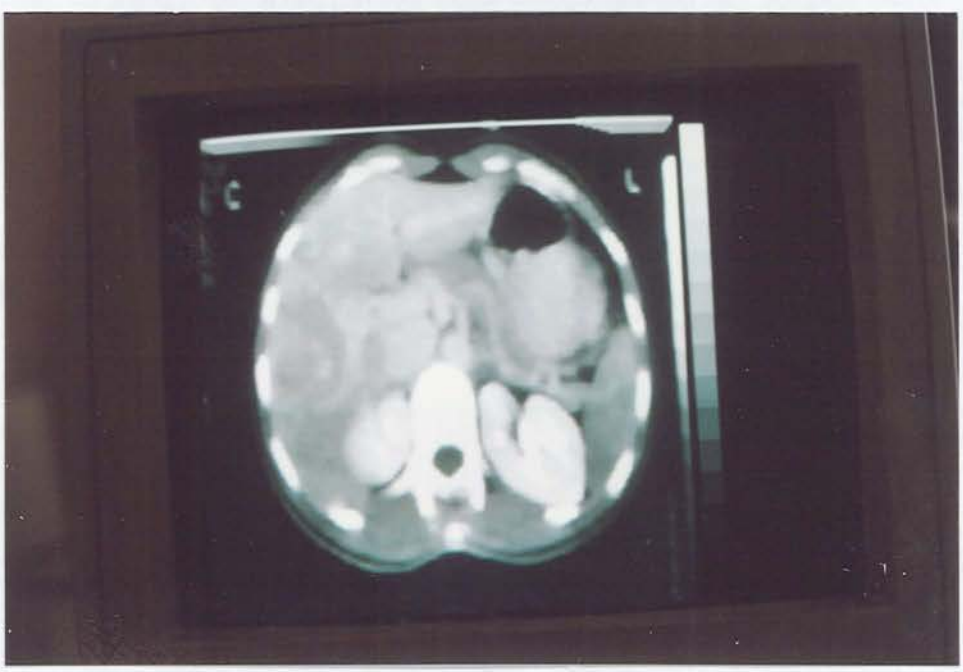


1. program 3*3 NEIGHBORHOOD AVERAGING filtering

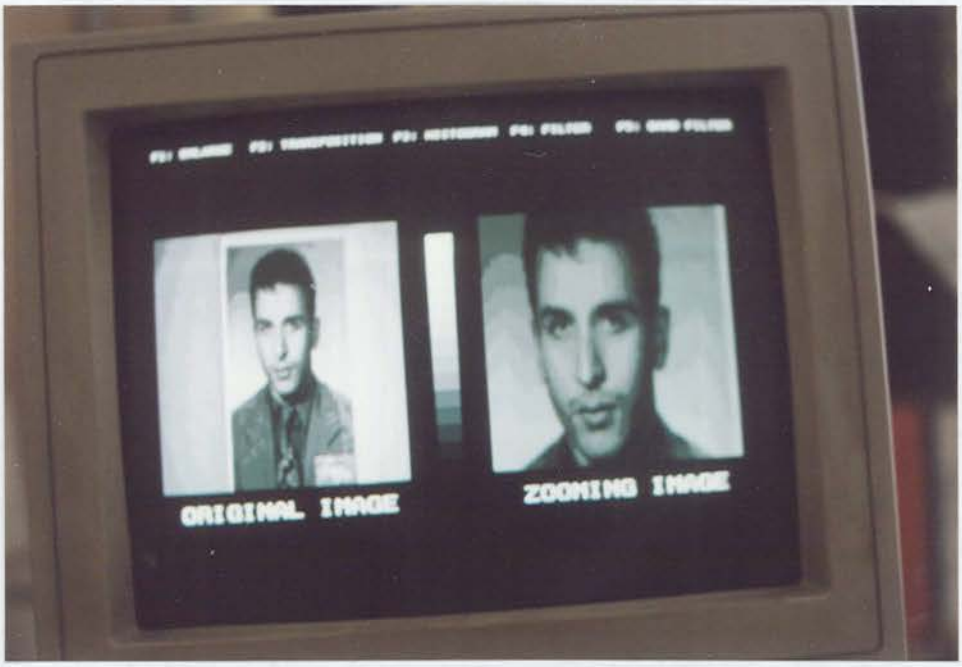


1. program 5*5 NEIGHBORHOOD AVERAGING filtering

1. program 3*3 MEDIAN FILTERING



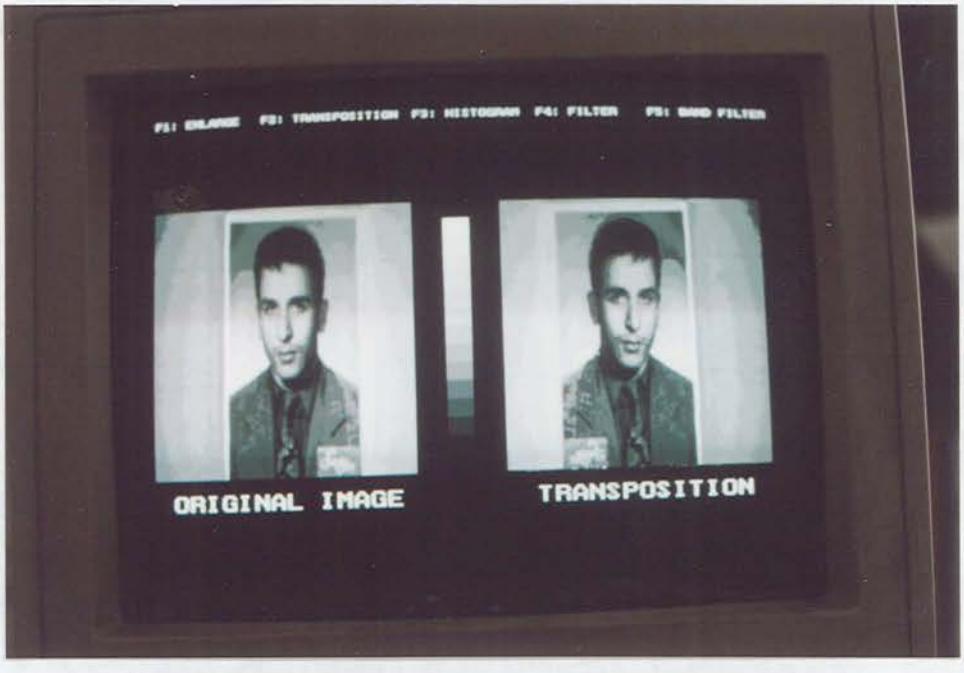
1. program 5*5 MEDIAN FILTERING



2. program X2 enlarge

2. program X4 enlarge

2. program UP/DOWN transposition



2. program LEFT/RIGHT transposition

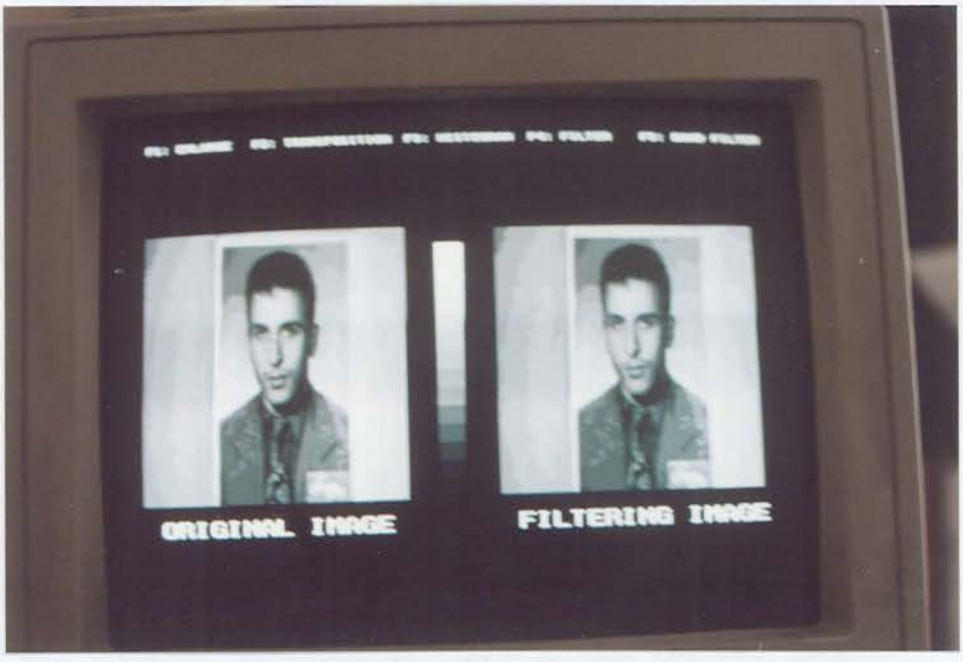
2. program X2 histogram

2. program X4 histogram

2. program ALL histogram

2. program FC1 filtering

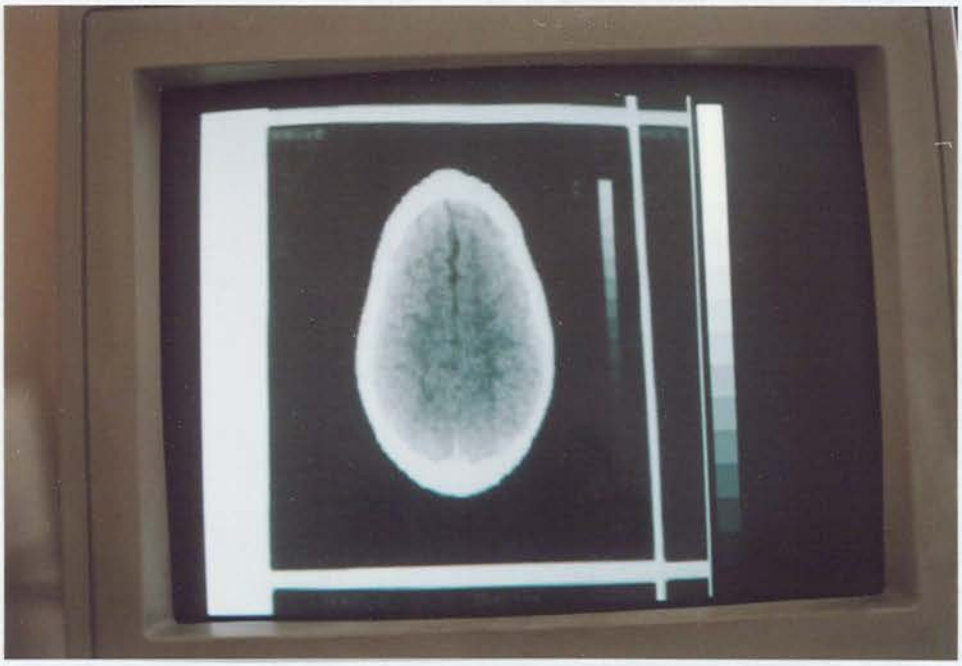
2. program FC2 filtering



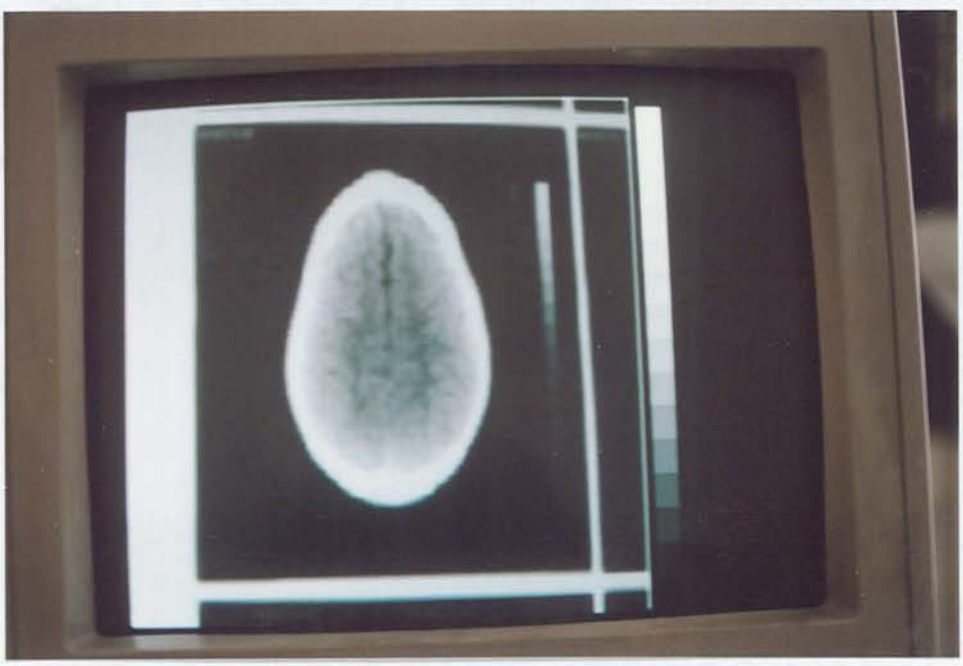
2. program FC3 filtering

2. program FC4 filtering

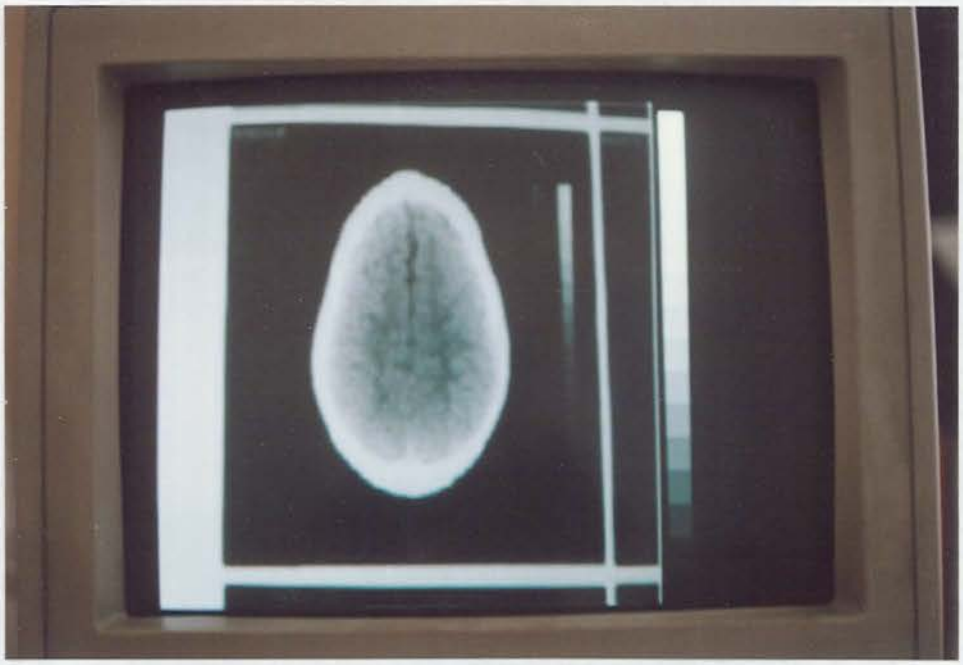
2. program BAND FILTER



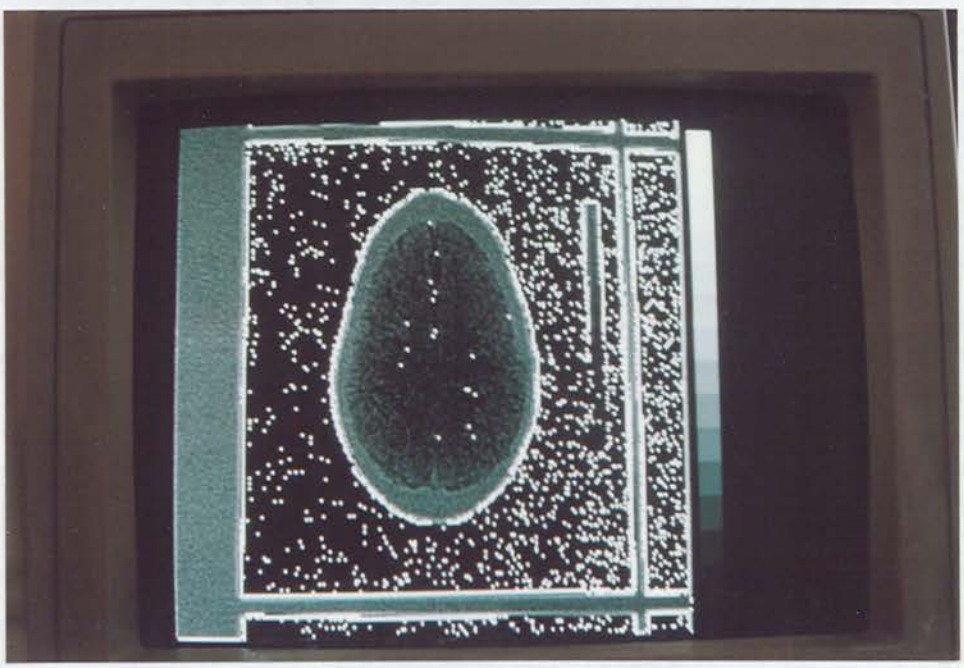
3. program orijinal görüntü



3. program HEAVY SMOOTHING filtering



3. program SMOOTHING filtering



3. program LIGHT EDGE ENHANCEMENT filtering

3. program EDGE ENHANCEMENT filtering

3. program HEAVY EDGE ENHANCEMENT filtering

2. program X4 enlargement

2. program BAND FILTER

EK 2

PROGRAM ROUTiNLERi

```
{$S-,R-,I-,V-,B-}

{*********************************************************}
{*                    GPMENU.PAS 4.02                    *}
{*********************************************************}

unit GpMenu;

interface

uses
  TPString,
  TPCrt,
  Gxmenu;

type

  MenuStatusType =              {Status of a menu operation}
   (MenuSuccess,                {Operation successful}
    MenuNoMem,                  {Insufficient memory}
    MenuFileNotFound,           {Menu library file not found}
    MenuNotLibraryFile,         {File is not a library file}
    MenuIdNotFound,             {Specified library element not
    found}
    MenuFileReadError,          {Error while reading menu file}
    MenuFileWriteError,         {Error while writing menu file}
    MenuFileCreationError,      {Unable to create library file}
    MenuFileCorrupt,            {Menu file is corrupt}
    MenuLibraryFull             {No room in library index to add
    a new entry}
    );

  MenuCharSet = set of Char; {User-defined keys to exit menu
selection}

  MenuKey = LongInt;            {What a menu selection returns
as identification}

  Orientation =                 {Which direction scrolling
  proceeds}
  (Vertical, Horizontal, NoOrient);

  FrameArray = array[FrameCharType] of Char; {Elements of a
window frame}

  MenuColorType =               {Colors used by the menu system}
   (FrameColor, HeaderColor, BodyColor, SelectColor,
  HiliteColor, HelpColor);
  MenuColorArray = array[MenuColorType] of Byte;

const
  {Tag denotes unframed submenus}
  LotusFrame = #255#255#255#255#255#255;
  NoFrame = LotusFrame;         {Synonym for LotusFrame}
  NoHelp = 0;                   {Help row to skip help
  altogether}
  FrameDelta : array[Boolean] of Byte = (1, 0);
```

```
  HideCursor : Boolean = True;  {False to leave hardware
  cursor on while menus displayed}


type
  Menu = ^MenuRec;
  ItemP = ^ItemRec;
  SubMenuP = ^SubMenuRec;
  MenuStackP = ^MenuStackRec;
  BufP = ^BufferArray;

  BufferArray = array[1..MaxInt] of Char;

  ItemRec =                          {27 bytes+name+help}
  record
    DisplayPos : Byte;               {Offset from top left corner of
menu for display}
    SelectPos : Byte;                {Byte in string to highlight and
cause selection, 0 for none}
    Key : MenuKey;                   {Key returned when item is
selected}
    Name : Pointer;                  {Pointer to string to display
for item}
    Help : Pointer;                  {Pointer to string to display
for item help}
    Next : ItemP;                    {Pointer to next item in list}
    Prev : ItemP;                    {Pointer to previous item in
    list}
    Sub : SubMenuP;                  {Pointer to submenu, nil if
    none}
    OnHeap : Boolean;                {True if name/help is allocated
    on heap}
  end;

  ItemList =
  record
    First : ItemP;                   {First item in menu}
    Last : ItemP;                    {Last item in menu}
    Current : ItemP;                 {Current item in menu}
  end;

  SubMenuRec =                       {43 bytes+header+screen buffers}
  record
    XL, YL : Byte;                   {Upper left corner of window
frame}
    XH, YH : Byte;                   {Actual bottom right corner of
window frame}
    YHelp : Byte;                    {Row where a help line starts}
    Orient : Orientation;            {Horizontal or vertical scroll}
    Frame : FrameArray;              {Characters for frame}
    Colors : MenuColorArray;         {Colors for parts of menu}
    LotusStyle : Boolean;            {True for menus without frames,
    ala Lotus}
    Header : Pointer;                {Title string for frame}
    Items : ItemList;                {Linked list of entries}
    Covers : BufP;                   {Points to buffer for screen
    covered by submenu}
```

```
    HelpCovers : BufP;          {Points to buffer for screen
    covered by help}
  end;

  MenuStackRec =
  record
    Top : SubMenuP;             {Points to active submenu}
    Next : MenuStackP;          {Remainder of the stack}
  end;

  MenuRec =
  record
    Root : SubMenuP;            {Root of menu}
    Active : SubMenuP;          {Currently active submenu}
    Stack : MenuStackP;         {Points to stack of active
    menus}
    UserFunc : Pointer;         {Points to user-supplied
    function}
    SelectKeys : MenuCharSet;  {User-defined keys to perform
    selection}
    Visible : Boolean;          {True when menus are onscreen}
  end;


procedure CheckMenuStatus(Mstatus : MenuStatusType);
  {-Check menu status, report and halt on any error}

function MenuStatus : MenuStatusType;
  {-Return status of previous operation}

function NewMenu(SelectKeys : MenuCharSet; UserFunc :
  Pointer) : Menu;
  {-Initialize a new menu system by returning a pointer to a
  new menu}

procedure SetMenuSelectKeys(Mnu : Menu; Skeys : MenuCharSet);
  {-Change the select key set of existing menu system as
specified}

procedure SubMenu(XL1, YL1, Yhelp1 : Byte;
                  Orient1 : Orientation;
                  Frame1 : FrameArray;
                  Colors1 : MenuColorArray;
                  HeaderStr : string
                  );
  {-Add a submenu to currently active item of currently
active submenu
    of currently active menu}

procedure PopSubLevel;
  {-Pop active menu from top of menu stack}

procedure MenuItem(NameStr : string; {Name of item}
                   DisplayPos1 : Byte; {Offset from upper
left corner of menu for item}
                   SelectPos1 : Byte; {Position within
namestr to hilite and select from}
```

```
                        Key1 : MenuKey; {Key to return when item
is selected}
                        HelpStr : string {Help string for item}
                        );
  {-Add an item to currently active submenu of currently
active menu,
    name space is allocated on heap}

procedure MenuItemPtr(NamePtr : Pointer; {Pointer to name of
                        item}
                        DisplayPos1 : Byte;
                        SelectPos1 : Byte;
                        Key1 : MenuKey;
                        HelpPtr : Pointer {Pointer to help for
                        item}
                        );
  {-Add an item to currently active submenu of currently
active menu,
    name space is NOT allocated on heap}

procedure DisposeMenu(Mnu : Menu);
  {-Dispose of all menu heap space}

procedure ResetMenu(Mnu : Menu);
  {-Set all selections to first item}

function MenuChoice(Mnu : Menu; var SelectKey : Char) :
  MenuKey;
  {-Display menu system, let user browse it, return menukey
  of selected item,
    return keystroke used to select item, leave menu on
screen}

procedure EraseMenu(Mnu : Menu; ResetSelections : Boolean);
  {-Erase active menus from the screen, reset selections to
base if desired}

procedure EraseMenuOntoStack(Mnu : Menu; var TStack :
  MenuStackP);
  {-Erase a menu system, saving the path of current selection
  on a stack}

procedure DrawMenuFromStack(Mnu : Menu; var TStack :
  MenuStackP);
  {-Draw a menu system using previously saved stack of items}

procedure WriteMenuLib(Mnu : Menu; Fname : string; ID :
  string);
  {-Write a menu system to a binary menu library}

function ReadMenuLib(Fname : string; ID : string; UserFunc :
  Pointer) : Menu;
  {-Read a menu system from a binary menu library}

procedure PackMenuLib(iname, oname : string);
  {-Remove obsolete menu entries from library iname, creating
oname}
```

```
   {_____________________________________________________________________
   _____________}
   {Following routines are primarily for internal use,
   interfaced for MAKEMENU}

procedure GClearWindow(XL, YL, XH, YH, Attr : Byte);
   {-Clear a region with specified attribute}

procedure DrawFrame(XL, YL, XH, YH, Attr : Byte; Frame :
   FrameArray);
   {-Draw a frame around a window}

procedure PushSubMenu(Mnu : Menu; SubMnu : SubMenuP);
   {-Put submenu onto active stack of the menu}

procedure PopSubMenu(Mnu : Menu);
   {-Remove submenu from active stack}

procedure DisposeSubMenu(var SubMnu : SubMenuP);
   {-Dispose of submenu and its children}

procedure DrawItem(SubMnu : SubMenuP; Item : ItemP; UserFunc
   : Pointer);
   {-Draw one item in a submenu}

procedure DrawSubMenu(SubMnu : SubMenuP; UserFunc : Pointer);
   {-Draw a submenu on-screen}

procedure EraseSubMenu(SubMnu : SubMenuP);
   {-Erase a submenu from the screen}
{=====================================================================
==============}

unit system3;
```

```
interface

usees crt,dos,printer,graph;

{-------------------------------------------------------------------------}

type
      fname =string[14];
      secenekler = array [1..16] of string[50];
      menusec = array [1..6,1..6] of string[12];
      dizi = array [1..5] of byte;
      flen =  array [1..23] of byte;
      ScreenType = array[1..25,1..80,0..1] of Byte;


 const  { special control keys }

      CR   : string[2] = 'CR';
    PgUp   : string[4] = 'PgUp';
    PgDn   : string[4] = 'PgDn';
    LEFT   : string[4] = 'LEFT';
    RIGHT  : string[5] = 'RIGHT';
    UP     : string[2] = 'UP';
    DOWN   : string[4] = 'DOWN';
    Home   : string[4] = 'Home';
    Esc    : string[3] = 'ESC';
    Ins    : string[3] = 'Ins';
    Del    : string[3] = 'Del';


    Letters : string[26] = 'abcdefghijklmnopqrstuvwxyz';
    Numbers : string[10] = '1234567890.';
    Specials: string[14] = '!"£$%^&*()_-+=:;,.<>@[]{}';
    MonoDisplay  = $B000;
    ColorDisplay = $B800;
    MaxScreen    =10;

    Firma : string[30]='YILMAZ Bilgisayar Muhendislik';

    Aylar : array[1..12] of string[7]=

('Ocak','Subat','Mart','Nisan','Mayis','Haziran','Temmuz ,
             'Agustos','Eylul','Ekim','Kasim','Aralik');


Var
     Screen :^ScreenType;
     Screens:Array[1..MaxScreen] of ^ScreenType;
     ScrpenStackPointer: 1..MaxScreen;
     zem,yaz:byte;
     chset : string ;
     key,k: char;
     mes,ipr :secenekler;
     opt : menusec;
     stat: fname;
     canc: boolean;
```

```
    VideoSeg : word;
    TestOkey: Boolean;

{-------------------------------------------------------------------------}


function valreal(a:string): Real;
procedure zemin(x:byte);
procedure yazi(x:byte);
procedure renk(x,y:integer);
procedure beep(a,b:integer);
function getkey:char;
procedure waitkey;
procedure waitEH;
procedure PrintWait(xp,yp :integer;mes:string);
procedure cursor(a:integer);
procedure CursorSize(x,y:Byte);
function strings(a,b:byte):string;
procedure ortala(b:string;cy:byte);
procedure pprint(x,y:byte;m:string);
procedure baslik(b:string);
procedure cerceve(b:string);
 procedure pencere(x1,y1,x2,y2,C,renkler:byte);
procedure perde(x1,y1,x2,y2:byte);
function getdaten:fname;
function gettime:fname;
procedure make_printer(wait:boolean);
function inputs(xp,yp,b:integer;a:string):string;
function inputi(xp,yp,b:integer;a:integer):Longint;
function inputr(xp,yp:integer;a:real;o,e:integer):real;
procedure clearinput;
procedure printparam(ss,xp,yp:byte; var m:secenekler);
procedure inputparam(ss,xp,yp:byte; var
m:secenekler;blen:flen);
procedure inputmesaj(m:secenekler;ss,xp,yp:byte);
procedure
form(m:secenekler;ss,xp,yp,wl:byte;b:string;fs:integer);
procedure copychrs(col,row,nrc:byte;var
bf:string;VideoSeg:word);
function upper(s:string):string;
procedure nop;
function lefts(a:string; n:integer):string;
function rights(a:string; n:integer):string;
procedure not_ready;
procedure ClrPos(xp,yp,nc:byte);
function ExistFile(path_and_filename:string):boolean;
procedure PrtSc(wait:boolean);
function Color(Zemin,Yazi:Byte):Byte;
procedure Condensed;
procedure NormalSize;
procedure ChangeScreen(ScNo:Byte);
procedure CopyScreen(ScNo:Byte);
procedure LoadScreen(ScNo:Byte);
procedure PushScreen;
procedure PopScreen;
Function Fill$(a:string; n:byte; f:char):String;
procedure TarihDegisikligi(x,y:word);
```

```
Procedure
menut(xt,yt,max,bnm,ty,FrameA,TextA,BarA:byte;tpc,Topic:strin
    g;
                              BB:boolean;var menuselect,xx:Shortint);

Procedure
ScrDown(UstSat,UstSut,AltSat,AltSut,SSay,Renk:Byte);
Procedure ScrUp(UstSat,UstSut,AltSat,AltSut,SSay,Renk:Byte);
{procedure calculator(x1,y1:byte);}
```

```
{$M 24096,0,655360}

USES SYSTEM3,CRT,GRAPH;

     CONST
          GRI : ARRAY [0..15] OF FILLPATTERNTYPE


     =((($00,$00,$00,$00,$00,$00,$00,$00),
       ($55,$55,$00,$00,$55,$55,$00,$00),
       ($00,$AA,$00,$AA,$00,$AA,$00,$AA),
       ($55,$55,$55,$00,$55,$55,$55,$00),
       ($CC,$33,$CC,$33,$CC,$33,$CC,$33),
       ($CC,$33,$CC,$33,$CC,$33,$CC,$33),
       ($99,$66,$99,$66,$99,$66,$99,$66),
       ($00,$FF,$00,$FF,$00,$FF,$00,$FF),
       ($AA,$55,$AA,$55,$AA,$55,$AA,$55),
       ($AA,$33,$AA,$55,$AA,$55,$AA,$55),
       ($66,$99,$66,$99,$66,$99,$66,$99),
       ($33,$CC,$33,$CC,$33,$CC,$33,$CC),
       ($AA,$AA,$AA,$FF,$AA,$AA,$AA,$AA),
       ($FF,$55,$FF,$55,$FF,$55,$FF,$55),
       ($AA,$AA,$FF,$FF,$AA,$AA,$FF,$FF),
       ($FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF));


     const

               GRAYSCALE : ARRAY [0..63] OF FILLPATTERNTYPE

                    = (   ($00,$00,$00,$00,$00,$00,$00,$00),     {
0  }
                          ($00,$00,$00,$08,$00,$00,$00,$00),     {
1  }
                          ($00,$00,$00,$10,$00,$00,$00,$00),     {
2  }
                          ($00,$00,$00,$1C,$00,$00,$00,$00),     {
3  }
                          ($00,$01,$00,$00,$01,$00,$00,$10),     {
27 }
                          ($00,$01,$00,$81,$00,$20,$00,$80),     {
5  }
                          ($00,$01,$00,$08,$00,$20,$00,$80),     {
4  }
                          ($00,$01,$00,$22,$00,$44,$00,$80),     {
6  }
                          ($00,$01,$00,$10,$00,$01,$00,$10),     {
22 }
                          ($00,$01,$00,$11,$00,$10,$00,$11),     {
25 }
                          ($00,$11,$00,$22,$00,$44,$00,$08),     {
7  }
                          ($A0,$00,$0A,$00,$A0,$00,$0A,$00),     {
```

```
36)
                ($00,$03,$00,$30,$00,$03,$00,$30),      {
21)
                ($00,$0A,$00,$A0,$00,$0A,$00,$A0),      {
20)
                ($00,$10,$00,$A5,$00,$01,$00,$5A),      {
11)
                ($00,$1A,$00,$A1,$00,$1A,$00,$A1),      {
19)
                ($11,$00,$EE,$00,$11,$00,$EE,$00),      {
34)
                ($00,$31,$00,$CE,$00,$31,$00,$CE),      {
29)
                ($00,$31,$00,$13,$00,$31,$00,$13),      {
28)
                ($00,$AA,$00,$55,$00,$AA,$00,$55),      {
16)
                ($00,$AA,$00,$AA,$00,$AA,$00,$AA),
(10 )
                ($00,$AA,$00,$55,$00,$AA,$00,$55),      {
12)
                ($33,$00,$CC,$00,$33,$00,$CC,$00),      {
35)
                ($01,$AA,$10,$AA,$01,$AA,$10,$AA),      {
26)
                ($00,$3A,$00,$A3,$00,$3A,$00,$A3),      {
23)
                ($11,$AA,$00,$55,$11,$AA,$00,$55),      {
30)
                ($00,$AF,$00,$FA,$00,$AF,$00,$FA),      {
18)
                ($00,$11,$00,$EE,$00,$11,$00,$EE),      {
15)
                ($00,$33,$00,$CC,$00,$33,$00,$CC),      {
14)
                ($00,$66,$00,$99,$00,$66,$00,$99),      {
13)
                ($00,$FF,$00,$77,$00,$FF,$00,$77),      {
17)
                ($0F,$F0,$0F,$F0,$0F,$F0,$0F,$F0),      {
9 )
                ($00,$EF,$00,$FE,$00,$EF,$00,$FE),      {
8 )
                ($55,$AA,$55,$AA,$55,$AA,$55,$AA),      {
32)
                ($00,$FF,$00,$FF,$00,$FF,$00,$FF),      {
24)
                ($00,$AA,$11,$00,$AA,$11,$00,$AA),      {
31)
                ($00,$AA,$FF,$55,$00,$AA,$FF,$55),      {
33)
                ($EE,$55,$E9,$55,$E9,$55,$E9,$55),      {
37)
                ($EE,$55,$E9,$55,$E9,$55,$EE,$55),      {
38)
                ($EE,$55,$EE,$55,$E9,$55,$EE,$55),      {
39)
```

```
                    ($EE,$55,$EE,$55,$EE,$55,$EE,$55),    {
40)
                    ($EB,$55,$EB,$55,$EB,$55,$EF,$55),    {
41)
                    ($FE,$55,$EB,$55,$EB,$55,$FE,$55),    {
42)
                    ($FE,$55,$FE,$55,$EB,$55,$FE,$55),    {
43)
                    ($FE,$55,$FE,$55,$FE,$55,$FF,$55),    {
44)
                    ($FF,$55,$FE,$55,$FE,$55,$FF,$55),    {
45)
                    ($FF,$55,$FE,$55,$FF,$55,$FF,$55),    {
46)
                    ($FF,$55,$FF,$55,$FF,$55,$FF,$55),    {
47)
                    ($FF,$55,$FF,$55,$FF,$55,$FF,$55),    {
48)
                    ($FF,$B8,$FF,$BB,$FF,$BB,$FF,$BA),    {
49)
                    ($FF,$BA,$FF,$B8,$FF,$BB,$FF,$BA),    {
50)
                    ($FF,$BA,$FF,$BB,$FF,$BA,$FF,$BA),    {
51)
                    ($FF,$BB,$FF,$BB,$FF,$BB,$FF,$BB),    {
52)
                    ($FF,$BA,$FF,$BA,$FF,$BA,$FF,$BB),    {
53)
                    ($FF,$BB,$FF,$BA,$FF,$BA,$FF,$BB),    {
54)
                    ($FF,$BB,$FF,$BA,$FF,$BA,$FF,$BA),    {
55)
                    ($FF,$BB,$FF,$BB,$FF,$BB,$FF,$BB),    {
56)
                    ($FF,$EE,$FF,$EE,$FF,$EE,$FF,$EF),    {
57)
                    ($FF,$EF,$FF,$EE,$FF,$EE,$FF,$EF),    {
58)
                    ($FF,$EF,$FF,$EF,$FF,$EF,$FF,$EF),    {
59)
                    ($FF,$FF,$E7,$EF,$E7,$FF,$FF,$FF),    {
60)
                    ($FF,$FF,$FF,$EF,$F7,$FF,$FF,$FF),    {
61)
                    ($FF,$FF,$FF,$FF,$EF,$FF,$FF,$FF),    {
62)
                    ($FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF)); {
63)

  type

   gray1=fillpatterntype;
```

```
  ImageArr = array[0..254,0..255] of byte;
  ImagePtr =^ImageArr;
  xxc =record
         x1,x2,x3,x4,x5,x6,x7,x8,x9:byte;
         end;
var
  Image   :  ImagePtr;
  FileName :  String[40];
  GraphDriver,GraphMode :  Integer;
  I,J,i1,j1 :  Word;
  xx,ade,xxc1,xxc2:byte;
  ImageFile :  File;
  dos:file of xxc;
  xd:xxc;
  Gray50,gray:gray1;
  gs:array[0..16] of gray1;



Function MedianFilter(a,b,c:byte;d:boolean):byte;


   Var
    Mat:Array[1..81] of byte;
    Ax,Ay,Aa,An:Byte;
    P:longint;
   Begin
    Case C of
     3 :  Begin Aa:=2;An:=5;End;
     5 :  Begin Aa:=3;An:=13;End;
     9 :  Begin Aa:=5;An:=41;End;
    End;
    If D then
     Begin
      For Ax:=1 to c do
       For Ay:=1 to c do
        Mat[Ax*Ay]:=Image^[A+Ay-Aa,B+Ax-Aa];
      For Ax:=1 to C*C-1 do
       For Ay:=Ax+1 to C*C do
        Begin
         P:=Mat[Ax];
         If P>Mat[Ay] then
          begin
           Mat[Ax]:=Mat[Ay];
           Mat[Ay]:=P;
          end;
        End;
       MedianFilter:=Mat[An];
      End
       Else
        Begin
         P:=0;
         For Ax:=1 to c do
          For Ay:=1 to c do
           P:=P+Image^[A+Ay-Aa,B+Ax-Aa];
           p:=p div (c*c);
         If Abs(Image^[a,b]-P)<10 THen MedianFilter:=P
```

```
            Else MedianFilter:=Image^[a,b];
        End;

    End;

Begin

    gray50[1]:=0;
    gray50[2]:=0;
    gray50[3]:=0;
    gray50[4]:=0;
    gray50[5]:=0;
    gray50[6]:=0;
    gray50[7]:=0;
    gray50[8]:=0;
     gs[0]:=gray50;

    gray50[1]:=0;
    gray50[2]:=0;
    gray50[3]:=0;
    gray50[4]:=$18;
    gray50[5]:=$18;
    gray50[6]:=0;
    gray50[7]:=0;
    gray50[8]:=0;
    gs[1]:=gray50;

    gray50[1]:=0;
    gray50[2]:=0;
    gray50[3]:=0;
    gray50[4]:=$3c;
    gray50[5]:=$3c;
    gray50[6]:=0;
    gray50[7]:=0;
    gray50[8]:=0;
    gs[2]:=gray50;

    gray50[1]:=0;
    gray50[2]:=0;
    gray50[3]:=$18;
    gray50[4]:=$18;
    gray50[5]:=$18;
    gray50[6]:=$18;
    gray50[7]:=0;
    gray50[8]:=0;
    gs[3]:=gray50;

    gray50[1]:=0;
    gray50[2]:=$18;
    gray50[3]:=$18;
    gray50[4]:=$18;
    gray50[5]:=$18;
    gray50[6]:=$18;
    gray50[7]:=$18;
    gray50[8]:=0;
    gs[5]:=gray50;
```

```
gray50[1]:=0;
gray50[2]:=0;
gray50[3]:=0;
gray50[4]:=$7e;
gray50[5]:=$7e;
gray50[6]:=0;
gray50[7]:=0;
gray50[8]:=0;
gs[4]:=gray50;

gray50[1]:=$18;
gray50[2]:=$18;
gray50[3]:=$18;
gray50[4]:=$18;
gray50[5]:=$18;
gray50[6]:=$18;
gray50[7]:=$18;
gray50[8]:=$18;
gs[7]:=gray50;

gray50[1]:=0;
gray50[2]:=0;
gray50[3]:=0;
gray50[4]:=$ff;
gray50[5]:=$ff;
gray50[6]:=0;
gray50[7]:=0;
gray50[8]:=0;
gs[6]:=gray50;

gray50[1]:=0;
gray50[2]:=0;
gray50[3]:=$3c;
gray50[4]:=$3c;
gray50[5]:=$3c;
gray50[6]:=$3c;
gray50[7]:=0;
gray50[8]:=0;
gs[8]:=gray50;

gray50[1]:=0;
gray50[2]:=0;
gray50[3]:=$7e;
gray50[4]:=$7e;
gray50[5]:=$7e;
gray50[6]:=$7e;
gray50[7]:=0;
gray50[8]:=0;
gs[9]:=gray50;

gray50[1]:=0;
gray50[2]:=$3c;
gray50[3]:=$3c;
gray50[4]:=$3c;
gray50[5]:=$3c;
gray50[6]:=$3c;
gray50[7]:=$3c;
```

```
gray50[8]:=0;
gs[10]:=gray50;


gray50[1]:=0;
gray50[2]:=0;
gray50[3]:=$ff;
gray50[4]:=$ff;
gray50[5]:=$ff;
gray50[6]:=$ff;
gray50[7]:=0;
gray50[8]:=0;
gs[11]:=gray50;

gray50[1]:=$3c;
gray50[2]:=$3c;
gray50[3]:=$3c;
gray50[4]:=$3c;
gray50[5]:=$3c;
gray50[6]:=$3c;
gray50[7]:=$3c;
gray50[8]:=$3c;
gs[12]:=gray50;

gray50[1]:=0;
gray50[2]:=$7e;
gray50[3]:=$7e;
gray50[4]:=$7e;
gray50[5]:=$7e;
gray50[6]:=$7e;
gray50[7]:=$7e;
gray50[8]:=0;
gs[13]:=gray50;

gray50[1]:=0;
gray50[2]:=$ff;
gray50[3]:=$ff;
gray50[4]:=$ff;
gray50[5]:=$ff;
gray50[6]:=$ff;
gray50[7]:=$ff;
gray50[8]:=0;
gs[14]:=gray50;

gray50[1]:=$7e;
gray50[2]:=$7e;
gray50[3]:=$7e;
gray50[4]:=$7e;
gray50[5]:=$7e;
gray50[6]:=$7e;
gray50[7]:=$7e;
gray50[8]:=$7e;
gs[15]:=gray50;

gray50[1]:=$ff;
gray50[2]:=$ff;
gray50[3]:=$ff;
```

```
     gray50[4]:=$ff;
     gray50[5]:=$ff;
     gray50[6]:=$ff;
     gray50[7]:=$ff;
     gray50[8]:=$ff;
     gs[16]:=gray50;


{    assign(dos,'renk3.ser');
   reset(dos);
 }  ClrScr;
   GetMem(Image,65280);{65280}
   Readln(FileName);
   Assign(ImageFile,FileName);
   Reset(ImageFile,65280);
   IF IORESULT=0 THEN
   BEGIN
   BlockRead(ImageFile,Image^,1);
   Close(ImageFile);
   END;
   DetectGraph(GraphDriver,GraphMode);
   GraphDriver:=9;
   GraphMode:=2;
   InitGraph(GraphDriver,GraphMode,'');
   ClearDevice;
   setgraphbufsize(65280);

INLINE($50/
       $53/
       $51/
       $52/
       $54/
       $55/
       $56/
       $57/
       $B4/$00/
       $80/$13/
       $CD/$10/
       $5F/
       $5E/
       $5D/
       $5C/
       $5A/
       $59/
       $5B/
       $58);



   j1:=1;
   For I:=1 To 253 Do
   begin
    i1:=1;

    For J:=1 To 253 Do
    begin
     xx:=MedianFilter(J,I,3,false);
```

```
    {  setfillpATTERN(GRIXX DIV 16),1);
      bar(j1*2,i1*2,(j1+1)*2,(i1+1)*2);}
{ xx:=xx div 8;
   for ade:=xx downto 0 do
    begin
     xxc1:=random(3);
     xxc2:=random(3);
     putpixel(j1+xxc1,i1+xxc2,1);
    end;  }


    putpixel(j1*2,i1*2,xx);
  inc(i1);



end;
 inc(j1);
end;









{  xx:=xx div 10;
seek(dos,xx);
read(dos,xd);

if xd.x1<>0 then
putpixel((i1-1),(j1-1),xd.x1);

if xd.x2<>0 then
putpixel((i1-1),(j1),xd.x2);
if xd.x3<>0 then
putpixel((i1-1),(j1+1),xd.x3);
if xd.x4<>0 then
putpixel((i1),(j1-1),xd.x4);
if xd.x5<>0 then
putpixel((i1),(j1),xd.x5);
if xd.x6<>0 then
putpixel((i1),(j1+1),xd.x6);
if xd.x7<>0 then
putpixel((i1+1),(j1-1),xd.x7);
if xd.x8<>0 then
putpixel((i1+1),(j1),xd.x8);
if xd.x9<>0 then
putpixel((i1+1),(j1+1),xd.x9);


  inc(i1,2);
```

```
    end;
    inc(j1,2);
    end;
      }

    WaitKey;
   End.


INLINE($50/
        $53/
        $51/
        $52/
        $54/
        $55/
        $56/
        $57/
        $B4/$00/
        $B0/$13/
        $CD/$10/
        $5F/
        $5E/
        $58/
        $5C/
        $5A/
        $59/
        $58/
        $58);



{$M 4096,0,655360}


USES CRT,GRAPH;

const  maxcolors=255;

 type

  ImageArr = array[0..254,0..255] of byte;
  ImagePtr =^ImageArr;
   Var ReelCord:word;
 var
  Image  : ImagePtr;
  FileName : String[40];
  GraphDriver,GraphMode : Integer;
  I,J,i1,j1,xxxx,yyyy,bb1,YY:word;
  xx,ade,xxc1,xxc2:byte;
  ImageFile : File;
  bb:byte;
  x1,x2,x3:byte;
  R,B,G:INTEGER;
```

```
Procedure PPixel(Row,Column:word;Color:byte);


   Begin
INLINE($50/
       $53/
       $51/
       $52/
       $54/
       $55/
       $56/
       $57/
       $B4/$0c/
       $B0/color/
       $B7/$2/
       $b9/Column/
       $ba/row/
       $CD/$10/
       $5F/
       $5E/
       $5D/
       $5C/
       $5A/
       $59/
       $5B/
       $58);
     End;

Procedure PuPixel(row,column,color:byte);

   const
      offset=$a000;
      EvenSegment=$0000;
      OddSegment=$7d00;
      IncValue=$140;



   Begin
     (If (row/2)=(row div 2) then
      begin
       reelcord:=row div 2;
       reelcord:=reelcord*incvalue+evensegment;
      end
       Else
        Begin
         reelcord:=Trunc(row/2);
         reelcord:=reelcord*incvalue+OddSegment;
        End;
        Mem[Offset:ReelCord+Column]:=Color;
        )
        ReelCord:=row*incvalue+evensegment+column;
        mem[offset:ReelCord]:=color;
    End;
```

```
Function MedianFilter(a,b,c:byte;d:boolean):byte;


  Var
   Mat:Array[1..81] of byte;
   Ax,Ay,Aa,An:Byte;
   P:longint;
  Begin
   Case C of
    3 : Begin Aa:=2;An:=5;End;
    5 : Begin Aa:=3;An:=13;End;        +
    9 : Begin Aa:=5;An:=41;End;
   End;
   If D then
    Begin
     For Ax:=1 to c do
      For Ay:=1 to c do
       Mat[Ax*Ay]:=Image^[A+Ay-Aa,B+Ax-Aa];
     For Ax:=1 to C*C-1 do
      For Ay:=Ax+1 to C*C do
       Begin
        P:=Mat[Ax];
        If P>Mat[Ay] then
         begin
          Mat[Ax]:=Mat[Ay];
          Mat[Ay]:=P;
         end;
        End;
       MedianFilter:=Mat[An];
      End
       Else
        Begin
         P:=0;
         For Ax:=1 to c do
          For Ay:=1 to c do
           P:=P+Image^[A+Ay-Aa,B+Ax-Aa];
           p:=p div (c*c);
         If Abs(Image^[a,b]-P)<10 THen MedianFilter:=P
          Else MedianFilter:=Image^[a,b];
        End;

    End;

Begin

  ClrScr;
  SetMem(Image,65280);{65280}
  Readln(FileName);
  Assign(ImageFile,FileName);
  Reset(ImageFile,65280);
  IF IORESULT=0 THEN
  BEGIN
  BlockRead(ImageFile,Image^,1);
  Close(ImageFile);
  END;
  detectgraph(graphdriver,graphmode);
```

```
    InitGraph(GraphDriver,GraphMode,'');
    ClearDevice;

    for i:=0 to 63 do
     setrgbpalette(i,i,i,i);

INLINE($50/
        $53/
        $B4/$10/
        $B0/$13/
        $B3/$00/
        $B7/$00/
        $CD/$10/
        $B4/$10/
        $B0/$13/
        $B3/$01/
        $B7/$00/
        $CD/$10/
        $5B/
        $58);

INLINE($50/
        $53/
        $E1/
        $52/
        $54/
        $55/
        $56/
        $57/
        $B4/$12/
        $B3/$33/
        $B0/$00/
        $CD/$10/
        $B4/$10/
        $B0/$1B/
        $BB/$0000/
        $B9/$ff00/
        $CD/$10/
        $5F/
        $5E/
        $5D/
        $5C/
        $5A/
        $59/
        $5B/
        $58);

INLINE($50/
        $53/
        $B4/$10/
        $B0/$03/
        $B3/$00/
        $CD/$10/
        $5B/
        $58);
```

```
    For I:=0 To 254 Do
    begin

     For J:=0 To 255 Do
     begin
       xx:=image^[i,j];
       putpixel(j,i,xx div 4);
      xx:=getpixel(j,i);
    end;
    end;
    REPEAT UNTIL KEYPRESSED;
   End.




{$M 6000,0,655360}


USES DOS,CRT,system3;


TYPE
     IMAGEARR=ARRAY[0..254,0..255] OF BYTE;
     IMAGEPTR=^IMAGEARR;

VAR
 I,J:WORD;
 DIRINFO:SEARCHREC;
 IMAGE:IMAGEPTR;
 FILENAME:STRING[12];
 DOSYA:FILE;
 ss:byte;
Procedure PuPixel(row,column,color:byte);

   const
       offset=$a000;
       EvenSegment=$0000;
       IncValue=$140;
   VAR REELCORD:WORD;

   Begin
        ReelCord:=row*incvalue+evensegment+column;
        mem[offset:ReelCord]:=color;
    End;



PROCEDURE SETGRAPH;

BEGIN
INLINE($50/
       $53/
       $51/
       $52/
       $54/
```

```
          $B4/$00/
          $B0/$13/
          $CD/$10/
          $5F/
          $5E/
          $5D/
          $5C/
          $5A/
          $59/
          $58/
          $58);
END;
{-------------------------------------------------------------------------------}


PROCEDURE GRAYLEVEL(X,Y:BYTE);
 BEGIN
   INLINE($50/
            $53/
            $51/
            $52/

            $B8/$10/$10/
            $B7/$00/
            $8A/$AE/X/
            $8A/$6E/Y/
            $8A/$DD/
            $8A/$E9/
            $8A/$F5/
            $CD/$10/

            $5A/
            $59/
            $5B/
            $58);

END;

{-------------------------------------------------------------------------------}

PROCEDURE SETTEXT;

BEGIN
INLINE($50/
          $53/
          $51/
          $52/
          $54/
          $55/
          $56/
          $57/
          $B4/$00/
          $B0/$03/
          $CD/$10/
```

```
        $5F/
        $5E/
        $5D/
        $5C/
        $5A/
        $59/
        $5B/
        $58);
END;
{-------------------------------------------------------------------------------}

PROCEDURE GRAY64(a:byte);

VAR BB:BYTE;

 BEGIN
  FOR BB:=0 TO 255 DO GRAYLEVEL(BB,BB div a);
 END;




BEGIN
 SETGRAPH;
    GRAY64(1);
 REPEAT
 FINDFIRST('*.IMG',ANYFILE,DIRINFO);

  WHILE DOSERROR=0 DO
   BEGIN
    FILENAME:=DIRINFO.NAME;
    GETMEM(IMAGE,65280);
    ASSIGN(DOSYA,FILENAME);
    RESET(DOSYA,65280);
    BLOCKREAD(DOSYA,IMAGE^,1);
    CLOSE(DOSYA);

    FOR I:=50 TO 250 DO
     FOR J:=1 TO 240 DO
      PUPIXEL(I-50,J,IMAGE^[I,J] DIV 4);



     FREEMEM(IMAGE,65280);
     FINDNEXT(DIRINFO);
   END;
  UNTIL KEYPRESSED;
   SETTEXT;
  END.



{$M 4096,0,655360}

USES SYSTEM3,CRT,GRAPH;
```

```
const   maxcolors=255;

 type

  ImageArr = array[0..254,0..255] of byte;
  ImagePtr =^ImageArr;
 var
  Image   : ImagePtr;
  FileName : String[40];
  GraphDriver,GraphMode : Integer;
  I,J,i1,j1 : Word;
  xx,ade,xxc1,xxc2:byte;
  ImageFile : File;
  pp:pointer;



 Function MedianFilter(a,b,c:byte;d:boolean):byte;


    Var
     Mat:Array[1..81] of byte;
     Ax,Ay,Aa,An:Byte;
     P:longint;
    Begin
     Case C of
      3 : Begin Aa:=2;An:=5;End;
      5 : Begin Aa:=3;An:=13;End;
      9 : Begin Aa:=5;An:=41;End;
     End;
     If D then
      Begin
       For Ax:=1 to c do
        For Ay:=1 to c do
         Mat[Ax*Ay]:=Image^[A+Ay-Aa,B+Ax-Aa];
       For Ax:=1 to C*C-1 do
        For Ay:=Ax+1 to C*C do
         Begin
          P:=Mat[Ax];
          If P>Mat[Ay] then
           begin
            Mat[Ax]:=Mat[Ay];
            Mat[Ay]:=P;
           end;
         End;
        MedianFilter:=Mat[An];
       End
        Else
         Begin
          P:=0;
          For Ax:=1 to c do
           For Ay:=1 to c do
            P:=P+Image^[A+Ay-Aa,B+Ax-Aa];
            p:=p div (c*c);
          If Abs(Image^[a,b]-P)<10 THen MedianFilter:=P
```

```
            Else MedianFilter:=Image^[a,b];
          End;

      End;

  Begin

    ClrScr;
    GetMem(Image,65280);{65280}
    Readln(FileName);
    Assign(ImageFile,FileName);
    Reset(ImageFile,65280);
    IF IORESULT=0 THEN
    BEGIN
    BlockRead(ImageFile,Image^,1);
    Close(ImageFile);
    END;

    GraphDriver:=9;
    GraphMode:=2;
    InitGraph(GraphDriver,GraphMode,'');
    ClearDevice;

INLINE($50/
       $53/
       $51/
       $52/
       $54/
       $55/
       $56/
       $57/
       $B4/$00/
       $B0/$13/
       $CD/$10/
       $5F/
       $5E/
       $5D/
       $5C/
       $5A/
       $59/
       $58/
       $58);




    j1:=1;
    For I:=1 To 84 Do
    begin
     i1:=1;

     For J:=1 To 126 Do
     begin
       xx:=MedianFilter(J*2,I*3,3,false);
       putpixel(i1*8,j1*4,xx);
     inc(i1);
```

```
    end;
     inc(j1);
    end;

  repeat
   WaitKey;
    if stat=cr then
     begin
      i:=imagesize(0,0,254,255);
      getmem(pp,i);
      getimage(0,0,254,255,pp^);
      cleardevice;
     end;
     if stat=up then
      begin
       putimage(0,0,pp,0);
       freemem(pp,i);
      end;

   until stat=esc;
  End.



{$M $2000,0,$F000}

USES CRT,DOS,SYSTEM3;

 TYPE SERIALPORT = RECORD
                    EDITORNAME:STRING[12];
                    BAUD_RATE:BYTE;
                    PARITYBIT:BYTE;
                    STOPBIT  :BYTE;
                    CHAR_SIZE:BYTE;
                   END;


 VAR MENUITEM,XX:SHORTINT;
     DOSYA:FILE OF SERIALPORT;
     DOSD :SERIALPORT;
     YY,SEND:BYTE;
     FILENAME:STRING[12];

PROCEDURE EDIT;

 BEGIN
  PUSHSCREEN;
  TEXTATTR:=$70;
  PPRINT(4,25,' ENTER EDITOR NAME : ');
  TEXTATTR:=$0F;
  DOSD.EDITORNAME:=INPUTS(25,25,12,DOSD.EDITORNAME);
  POPSCREEN;
 END;

PROCEDURE BAUDRATE;
```

```
 BEGIN
  PUSHSCREEN;
  XX:=DOSD.BAUD_RATE+1;
  MENUT(48,14,8,20,1,$70,$70,$07,
         '110 bits/sec-150 bits/sec-300 bits/sec-600
bits/sec-1200 bits/sec-2400 bits/sec-4800 bits/sec-9600
bits/sec-',
         'BAUD RATE',TRUE,MENUITEM,XX);
    IF MENUITEM<>-1 THEN DOSD.BAUD_RATE:=MENUITEM-1;
  POPSCREEN;
  MENUITEM:=0;
 END;

PROCEDURE PARITY;

 BEGIN
  PUSHSCREEN;
  IF DOSD.PARITYBIT=1 THEN XX:=1 ELSE XX:=2;
  MENUT(48,14,3,20,1,$70,$70,$07,
         'NON PARITY-ODD PARITY-EVEN
PARITY-','PARITY',TRUE,MENUITEM,XX);
    IF MENUITEM<>-1 THEN
     BEGIN
      IF MENUITEM=1 THEN DOSD.PARITYBIT:=0;
      IF MENUITEM=2 THEN DOSD.PARITYBIT:=1;
      IF MENUITEM=3 THEN DOSD.PARITYBIT:=3;
     END;
  POPSCREEN;
  MENUITEM:=0;
 END;

PROCEDURE STOPBITS;

 BEGIN
 PUSHSCREEN;
 IF DOSD.STOPBIT=0 THEN XX:=1 ELSE XX:=2;
  MENUT(48,14,2,20,1,$70,$70,$07,
         'ONE BIT-TWO BITS-','STOP BITS',TRUE,MENUITEM,XX);
    IF MENUITEM<>-1 THEN
     IF MENUITEM=1 THEN DOSD.STOPBIT:=0 ELSE DOSD.STOPBIT:=3;
  POPSCREEN;
  MENUITEM:=0;

 END;

PROCEDURE CHARSIZE;

 BEGIN
 PUSHSCREEN;
 XX:=DOSD.CHAR_SIZE-1;
  MENUT(48,14,2,20,1,$70,$70,$07,
         'SEVEN BITS-EIGHT BITS-','CHARACTER
SIZE',TRUE,MENUITEM,XX);
    IF MENUITEM<>-1 THEN DOSD.CHAR_SIZE:=MENUITEM+1;
  POPSCREEN;
  MENUITEM:=0;
```

```
  END;

PROCEDURE EDITOR;

  BEGIN
  PUSHSCREEN;
  TEXTATTR:=$70;
  PPRINT(4,25,'ENTER FILE NAME: ');
  TEXTATTR:=$0F;
  FILENAME:=INPUTS(21,25,12,FILENAME);
  SWAPVECTORS;
  EXEC(DOSD.EDITORNAME,FILENAME);
  SWAPVECTORS;
  POPSCREEN;
  END;

PROCEDURE SENDDATA;


  PROCEDURE DATASEND(XY:BYTE);

    BEGIN
     ASM
      MOV AH,$01
      MOV AL,XY
      MOV DX,$0
      INT $14
     END;
    END;



  VAR X1,X2,X3:BYTE;
      DOSYA1:TEXT;
      STRIN:STRING;
      SS:STRING[1];
  BEGIN
   PUSHSCREEN;
   DOSD.BAUD_RATE:=DOSD.BAUD_RATE AND $07;
   DOSD.PARITYBIT:=DOSD.PARITYBIT AND $03;
   DOSD.STOPBIT:=DOSD.STOPBIT AND $01;
   DOSD.CHAR_SIZE:=DOSD.CHAR_SIZE AND $03;
   X1:=DOSD.BAUD_RATE SHL 5;
   X2:=DOSD.PARITYBIT SHL 3;
   X3:=DOSD.STOPBIT SHL 2;
   SEND:=X1 OR X2 OR X3 OR DOSD.CHAR_SIZE;
    ASM
     MOV AH,$00
     MOV AL,SEND
     MOV DX,$0000
     INT $14
    END; {END OF ASM}
    ASSIGN(DOSYA1,FILENAME);
    IF EXISTFILE(FILENAME) THEN
     BEGIN
      RESET(DOSYA1);
      WHILE NOT(EOF(DOSYA1)) DO
```

```
        BEGIN
         READLN(DOSYA1,STRIN);
         X1:=BYTE(STRIN[0]);
         FOR X2:=1 TO X1 DO
          BEGIN
           X3:=BYTE(STRIN[X2]);
           DATASEND(X3);
          END;
         END;
         X3:=$0D;
         DATASEND(X3);
         PPRINT(4,25,'SENDING END.');
         WAITKEY;
         CLOSE(DOSYA1);
       END
       ELSE
        BEGIN
         TEXTATTR:=$F0;
         PPRINT(4,25,FILENAME+' NOT FOUND');
         WAITKEY;
         TEXTATTR:=$1E;
        END;
    POPSCREEN;
  END;


PROCEDURE SETUP;

  BEGIN



    PUSHSCREEN;
    YY:=1;
    REPEAT
    XX:=YY;
    MENUT(34,12,6,25,1,$70,$70,$07,
          'EDITOR-BAUD RATE-PARITY-STOP BITS-CHARACTER
SIZE-SAVE- ',
          'SETUP',TRUE,MENUITEM,XX);
    YY:=XX;
     CASE MENUITEM OF
       1 : EDIT;
       2 : BAUDRATE;
       3 : PARITY;
       4 : STOPBITS;
       5 : CHARSIZE;
     END;
    UNTIL (MENUITEM=-1) OR (MENUITEM=6);
    IF MENUITEM=6 THEN
     BEGIN
      ASSIGN(DOSYA,'SETUP.SER');
      REWRITE(DOSYA);
      WRITE(DOSYA,DOSD);
     END;
    POPSCREEN;
```

```
            'DOWN B    ','DOWN T    ','C TIME    ','S TIME    ','S
UP B    ','S UP T    ','S DOWN B ',
            'S DOWN T ','NORMAL B ','NORMAL T ');


TYPE DEGER=ARRAY[1..200] OF STRING[70];
     SSATIR=STRING[70];

VAR SATIR:STRING[60];
    TAMDEGER:DEGER;
    TAMSATIR:SSATIR;
    I:BYTE;
    MENU,B2:SHORTINT;
    FILENAME:STRING[12];
    DOSYA:FILE OF SSATIR;
    DOSDEGER:BOOLEAN;
    ASILDEGER:BYTE;
    KUTUKSONU:BYTE;
    EKRANYERI:BYTE;

PROCEDURE SCROOLUP(A1,A2,A3,A4,A5,A6:BYTE);

 BEGIN
  ASM
   MOV AH,06
   MOV CL,A1
   MOV CH,A2
   MOV DL,A3
   MOV DH,A4
   MOV AL,A5
   MOV BH,A6
   INT $10
  END;
END;
PROCEDURE SCROOLDOWN(A1,A2,A3,A4,A5,A6:BYTE);

 BEGIN
  ASM
   MOV AH,07
   MOV CL,A1
   MOV CH,A2
   MOV DL,A3
   MOV DH,A4
   MOV AL,A5
   MOV BH,A6
   INT $10
  END;
 END;


PROCEDURE EDITOR(XX:BYTE);


 VAR J,K,KK:BYTE;
     SS:STRING;

 BEGIN
```

```
I:=KUTUKSONU;J:=KUTUKSONU;K:=EKRANVERI;
REPEAT
 IF I=J THEN
  BEGIN
   IF (DOSDEGER=TRUE) AND (I<=KUTUKSONU) THEN
    BEGIN
     FILLCHAR(SATIR,SIZEOF(SATIR),0);
     TAMSATIR:=TAMDEGER[J];
     KK:=BYTE(TAMSATIR[0])-9;
     SATIR[0]:=CHAR(KK);
     FOR KK:=1 TO BYTE(TAMSATIR[0])-9 DO
     SATIR[KK]:=TAMSATIR[KK+9];
    END
   ELSE
    BEGIN
     FILLCHAR(TAMSATIR,SIZEOF(TAMSATIR),0);
     TAMSATIR:=PARA[XX];
     FILLCHAR(SATIR,SIZEOF(SATIR),0);
    END;
  END;
 IF J<I THEN
  BEGIN
   FILLCHAR(SATIR,SIZEOF(SATIR),0);
   TAMSATIR:=TAMDEGER[J];
   KK:=BYTE(TAMSATIR[0])-9;
   SATIR[0]:=CHAR(KK);
   FOR KK:=1 TO BYTE(TAMSATIR[0])-9 DO
   SATIR[KK]:=TAMSATIR[KK+9];
  END;
  TEXTATTR:=$31;
   PPRINT(6,K+3,TAMSATIR);
   SATIR:=INPUTS(15,K+3,60,SATIR);
   SS:='';
   FOR KK:=1 TO 9 DO SS:=SS+TAMSATIR[KK];
   SS:=SS+SATIR;
   TAMDEGER[J]:=SS;
   IF STAT=UP THEN
    BEGIN
     IF K>1 THEN
      BEGIN
       DEC(K);
       DEC(J);
      END
       ELSE
        BEGIN
         IF J>1 THEN
          BEGIN
           DEC(J);
           SCROOLDOWN(5,3,75,22,1,$1E);
          END;
        END;
    END;
  IF (STAT=DOWN) AND (J<I) THEN
    BEGIN
     IF K<20 THEN
      BEGIN
         INC(K);
```

```
  PUSHSCREEN;
   MENUT(64,15,7,10,1,$70,$70,$07,
          'FLASH-THROW-UP-DOWN-STAR UP-STAR DOWN-NORMAL-',
          'BOLD',TRUE,MENU,B2);
    CASE MENU OF

      1 : ASILDEGER:=2;
      2 : ASILDEGER:=4;
      3 : ASILDEGER:=6;
      4 : ASILDEGER:=8;
      5 : ASILDEGER:=12;
      6 : ASILDEGER:=14;
      7 : ASILDEGER:=16;
     END;
    POPSCREEN;
    END;

 IF STAT='F3' THEN
  BEGIN
   PUSHSCREEN;
   MENUT(64,15,7,10,1,$70,$70,$07,
         'FLASH-THROW-UP-DOWN-STAR UP-STAR DOWN-NORMAL-',
         'THIN',TRUE,MENU,B2);

   CASE MENU OF
      1 : ASILDEGER:=3;
      2 : ASILDEGER:=5;
      3 : ASILDEGER:=7;
      4 : ASILDEGER:=9;
      5 : ASILDEGER:=13;
      6 : ASILDEGER:=15;
      7 : ASILDEGER:=17;
   END;
   POPSCREEN;
  END;

 IF STAT='F4' THEN
  BEGIN
   ASILDEGER:=10;
  END;

 IF STAT='F5' THEN
  BEGIN
   ASILDEGER:=11;
  END;

 IF STAT='F6' THEN
  BEGIN
   PUSHSCREEN;

   POPSCREEN;
  END;

IF STAT='F1' THEN
 BEGIN

 END;
```

```
  IF STAT='F7' THEN EDITOR(ASILDEGER);

  IF STAT='F8' THEN
   BEGIN
    IF KUTUKSONU>1 THEN
     BEGIN
      ASSIGN(DOSYA,FILENAME);
      REWRITE(DOSYA);
      I:=1;
      REPEAT
       TAMSATIR:=TAMDEGER[I];
       WRITE(DOSYA,TAMSATIR);
       INC(I);
      UNTIL I=KUTUKSONU;
      CLOSE(DOSYA);
     END;
   END;
 UNTIL STAT=ESC;
 END;




BEGIN
 IF PARAMCOUNT <> 0 THEN
   FILENAME:=PARAMSTR(1)
  ELSE
   BEGIN
    TEXTATTR:=$F0;
    PPRINT(10,10,'ENTER FILE NAME :');
    FILENAME:=INPUTS(27,10,12,'');
   END;
 FILLCHAR(TAMDEGER,SIZEOF(TAMDEGER),0);
 ASSIGN(DOSYA,FILENAME);
 IF EXISTFILE(FILENAME) THEN
  BEGIN
   DOSDESER:=TRUE;
   KUTUKSONU:=1;
   EKRANYERI:=1;

   RESET(DOSYA);
   I:=1;
   WHILE NOT(EOF(DOSYA)) DO
    BEGIN
     READ(DOSYA,TAMSATIR);
     TAMDEGER[I]:=TAMSATIR;
     INC(I);
    END;
    CLOSE(DOSYA);
    KUTUKSONU:=I-1;
    EKRANYERI:=1;
  END
   ELSE
    BEGIN
```

```
      DOSDEGER:=FALSE;
      KUTUKSONU:=1;
      EKRANYERI:=1;
     END;
ASILDEGER:=17;
BASLIKAT;
GIRIS;
(I:=1;
REPEAT
S:=INPUTS(9,I,70,'');
INC(I);
UNTIL I=24; )
END.


{$M 4096,0,655360}


USES SYSTEM3,CRT,GRAPH;

const  maxcolors=255;

 type

  ImageArr = array[0..234,0..255] of byte;
  ImagePtr =^ImageArr;
 var
  Image  :  ImagePtr;
  FileName : String[40];
  GraphDriver,GraphMode :  Integer;
  I,J,ii,jj : Word;
  xx,ade,xxc1,xxc2:byte;
  ImageFile : File;
  pp:pointer;



Procedure PuPixel(row,column,color:byte);

  const
     offseet=$a000;
     EvenSegment=$0000;
     OddSegment=$7d00;
     IncValue=$140;

  Var ReelCord:word;

  Begin
   If (row/2)=(row div 2) then
    begin
     reelcord:=row div 2;
     reelcord:=reelcord*incvalue+evensegment;
    end
     Else
      Begin
       reelcord:=Trunc(row/2);
       reelcord:=reelcord*incvalue+OddSegment;
```

```
      End;
      Mem[Offset:ReelCord]:=Color;
    End;



Function MedianFilter(a,b,c:byte;d:boolean):byte;


   Var
    Mat:Array[1..81] of byte;
    Ax,Ay,Aa,An:Byte;
    P:longint;
   Begin
    Case C of
     3 : Begin Aa:=2;An:=5;End;
     5 : Begin Aa:=3;An:=13;End;
     9 : Begin Aa:=5;An:=41;End;
    End;
    If D then
     Begin
      For Ax:=1 to c do
       For Ay:=1 to c do
        Mat[Ax*Ay]:=Image^[A+Ay-Aa,B+Ax-Aa];
      For Ax:=1 to C*C-1 do
       For Ay:=Ax+1 to C*C do
        Begin
         P:=Mat[Ax];
         If P>Mat[Ay] then
          begin
           Mat[Ax]:=Mat[Ay];
           Mat[Ay]:=P;
          end;
         End;
        MedianFilter:=Mat[An];
       End
        Else
         Begin
          P:=0;
          For Ax:=1 to c do
           For Ay:=1 to c do
            P:=P+Image^[A+Ay-Aa,B+Ax-Aa];
            p:=p div (c*c);
          If Abs(Image^[a,b]-P)<10 THen MedianFilter:=P
           Else MedianFilter:=Image^[a,b];
         End;

     End;

  Begin

   ClrScr;
   GetMem(Image,65280);{65280}
   Readln(FileName);
   Assign(ImageFile,FileName);
   Reset(ImageFile,65280);
```

```
    IF IORESULT=0 THEN
    BEGIN
    BlockRead(ImageFile,Image^,1);
    Close(ImageFile);
    END;

    GraphDriver:=9;
    GraphMode:=2;
    InitGraph(GraphDriver,GraphMode,'');
    ClearDevice;

INLINE($50/
       $53/
       $51/
       $52/
       $54/
       $55/
       $56/
       $57/
       $B4/$00/
       $30/$13/
       $CD/$10/
       $5F/
       $5E/
       $5D/
       $5C/
       $5A/
       $59/
       $58/
       $58);




    For I:=1 To 200 Do
    begin

     For J:=1 To 200 Do
     begin
     I1:=trunc(i*1.25);
     j1:=trunc(j*1.25);
       xx:=MedianFilter(J1,i1,3,true);
       putpixel(i*4,j,xx);
    end;
    end;

   repeat
    WaitKey;
     if stat=cr then
      begin
       i:=imagesize(0,0,254,255);
       getmem(pp,i);
       getimage(0,0,254,255,pp^);
       cleardevice;
      end;
      if stat=up then
```

```
      begin
       putimage(0,0,pp,0);
       freemem(pp,i);
      end;

 until stat=esc;
End.
```