YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Görüntü ve Şekil Tanıma

YÜKSEK LİSANS TEZİ

Meir Eskinazi

1989

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ OKUMA VE ŞEKİL TANIMA

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜH. MEİR ESKİNAZİ

İSTANBUL, 1989

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ OKUMA VE ŞEKİL TANIMA

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜH. MEİR ESKİNAZİ

İSTANBUL, 1989

ÖNSÖZ

Dijital Bilgisayarların gelişmesi, her konudaki uygulamaları günden güne arttırmaktadır. Buna insanlardaki, bilgisayarlara, daha önce diğer makinalara yaptırılamamış şeyleri yaptırma güdüsü de yardım etmektedir. Bunlar arasında konuşma, ses ve şekil tanıma gibi insanların gündelik fonksiyonları önemli yer tutmaktadır. Fakat Bilgisayar alanındaki tüm gelişmelere rağmen ışığı, sesi, ısıyı, elektriksel sinyallere dönüştüren TRANSDUCER'lar henüz istenen kesin başarıya ulaşamamışlardır. Hala algılama problemleri ile bizleri yanıltabilmektedir.

Bu nedenle 1960'lı yılların başından beri araştırmacıların dikkatini genişçe bir şekilde çeken görüntü tanıma konusu iyi bir çözümden henüz uzaktır. Düzenli olarak artan araştırma ve geliştirme çalışmalarına rağmen hata oranı insanınkiyle kıyaslandığında çok daha yüksektir.

Bu projede amaç, karakter fontlarını ve el yazısını okumak, işlemek, harf kataloğu oluşturmak ve bunun yardımıyla bir dijital şekilde hangi harfin bulunduğunu tahmin etmektir.

Bu projenin hazırlanmasında değerli yardımlarından dolayı Sayın Prof. M.Yahya KARSLIGİL'e ve Sayın Doç. Kirkor HARUTUNYAN'a, tezimin yazılmasında emeği geçen tüm arkadaşlarıma teşekkür ederim.

YILDIZ ÜNİVERSİTESİ Meir ESKİNAZİ

İSTANBUL, 1989

İÇİNDEKİLER

3. GÖRÜNTÜ TANIMA

ÖZET

Bu güne kadar görüntü tanıma konusundaki çalışmalara Bilgisayar bilimlerinin yanısıra İstatistik, Yöneylem Araştırması, Haberleşme, Kontrol, Biyoloji, Psikoloji ve Dil Bilimleri disiplinlerinden de geniş katkı sağlanmıştır. Fakat her görüntü tanıma uygulaması kendi özel karakteristiklerine sahip olduğu için bu değişik dallardan, değişik şekil ve ölçülerde yararlanmaktadır ve henüz çalışma alanından bağımsız yöntemler geliştirilememiştir.

Bunlara rağmen istatistiksel karar verme teorileri her zaman baş vurulması gereken bir yardımcı olmuştur. Zira görüntüler özal tanıtıcı vektörler haline yorumlandıktan sonra bu bize matematiksel (doğal olarak bilgisayar uygulamalarına en uygun bir şekil olan) prosedürler sağlamaktadır.

Domen'e olan bağımlılık araştımacıları, kesin sonuca ulaşabilmek için farklı hipotezleri, farklı bilgisayar konfigürasyonlarında, farklı uygulamalar için denemeye yöneltmiştir.

Görüntü Okuma ve Şablon Eşleme yöntemiyle şekil tanıma olan bu projede ana hedef, dijital bir görüntünün, şekil tarayıcıdan alınarak işlenmesi ve önceden oluşturulmuş cisim kataloğu içinde hangisine en çok benzediğinin bulunmasıdır. Sistem belli başlı dört bölümden meydana gelir; Görüntü Okuma-Yaratma, Görüntü İşleme, Şekil Tanıma ve Öğrenme.

Birinci bölümde görüntü tarayıcı yardımıyla bir sayfanın dijital hale getirilmesi yada mouse aracılığıyla ekran üzerinde bir şekil yaratılması bulunmaktadır.

İkinci bölümde ekrandaki bir şeklin filtrelenmesi yada ölçeğinin değiştirilmesi gibi görüntü işleme fonksiyonları bulunmaktadır.

Tanıma bölümünde TEMPLATE MATCHING yöntemiyle karakter tanınması bulunmaktadır. Ayrıca şablonların kontrol edilmesi veya silinmesi gibi database kontrol fonksiyonları da bu bölüme girmektedir.

Son bölüm olan öğrenmede, tanınamayan şekillerin yada şüpheli olan sonuçların diskete kaydedilmesi bulunmaktadır.

ABSTRACT

Upto now, the extensive contributions for the studies about image recognation which have been supplied from not only the computer sciences but from Statistics, Operational Research, Communication, Control and the diciplines of Biology, Psychology and Linguistics as well. Inspite of this due to each image recognation application has their individual characteristics, image recognation benefits from these branches in terms of different ways and dimensions.

Inspite of the existence of these things, statistical decision the ories always have been the helper that should be applied. Since, after the images have been formed into the special significant vectors this provides us mathematical procedures which are the optimum methods for the computer applications.

Dependency on domain, leads researchers on trying the different hypothesis in different configurations for different applications to attain certain results.

The main aim in this project which is about the recognation of pattern by applying the Image Reading and Pattern Matching methods is, processing the digital image by taking it from scanner and to be found that which of them, it has matched. The system consists of four part; Image reading-generation, image processing, image recognation and learning.

In the first part, It is found that digitisation or generation an image on the screen in terms of mouse.

In the second part, some processing functions are found such as filtration of the image which is on the screen or conversion of the dimension.

In the third part, the recognation of the character by means of TEMPLATE MATCHING method is found. In addition to this, the database control functions such as control or deletion of the patterns are discussed in this part.

In the last part, there are procedures which record unrecognizable patterns or suspicious results on the diskette.

1

GÖRÜNTÜ TARAYICI

Canon IX-8 Image Scanner/Interface IBM PC uyumlu bilgisayarlarda görüntü okuma ve işleme konusunda geniş imkanlar sağlayan bir sistemdir. Aslında bir FAXIMILE karakterine sahip sayfa tarayıcıdan, bunun PC ile haberleşmesini sağlayan bir arabirimden ve tüm bunların kullanılmasını sağlayan bir yazılımdan oluşur. Bu yazılım sayfa tarayıcının programlanmasını ve PC ile haberleşmesini kontrol eder ve diğer tüm dillerden kullanılabilen dokuz fonksiyon sağlar. Intel 8086 assembler dili ile hazırlanmıştır. Bu program belleğe yüklendikten sonra software interrupt aracılığıyla kullanılır.

Görüntü tarayıcının teknik özelliklerine kısaca bir göz atarsak. Canon IX-8, 2048 görüntü elemanlı (pixel) bir okuyucu (CCD- Charge Coupled Device) kullanmaktadır. Bu okuyucu ile taranan görüntü bilgisayarın hafızasına

aktarılmakta ve yardımcı programları aracılığıyla bunun üzerinde herhangi bir değişiklik yapılabilmektedir. IBM uyumlu her türlü bilgisayar ile çalışabilmektedir. Kabul edebildiği kağıt genişliği 297 mm'dir. Okuyabildiği görüntü genişliği ise 208 mm'dir. Yatay tarama yoğunluğu 8 nokta/mm ve dikey tarama yoğunluğu 7.7 veya 3.85 nokta/mm'dir. Tarama hızı 7.7 nokta/mm de 7 saniye/sayfa ve 3.85/mm de 3.5 saniye/sayfa'dır. Bilgisayara bağlantı, 14 bacak amphenol tip konnektör'e sahip bir paralel arabirim yardımıyla olmaktadır ve bunun iletişim hızı 1.33 Mbit/saniye'dir.

Görüntü Tarayıcı Yardımcı Program Rutinleri

1- Interface kartının kontrol edilmesi:

Giriş : AH=0 (fonksiyon kodu)

Fonksiyon : Görüntü Tarayıcı kartının takılı olup olmadığını test eder.

Çıkış : AX>=0 : Kart takılı.

AX=-1 : Kart takılı değil.

2- Görüntü okuma parametrelerinin kontrol edilmesi:

Giriş : AH=1 (fonksiyon kodu)

AL=00XX00YY okuma ölçeklendirme faktörü

XX:Dikey incelik

00:1/1 (8 dot/mm , 1664 dot)

01:1/2 (4 dot/mm , 832 dot)

10:1/4 (2 dot/mm , 416 dot)

YY:Yatay incelik

00:1/1 (7.7 dot/mm , 2286 dot)

01:1/2 (3.85 dot/mm , 1143 dot)

10:1/4 (1.925 dot/mm , 571 dot)

BX=Dikey tarama başlangıç noktası.

CX=Taranacak nokta sayısı.(0 en sağ ucu belirtir)

DX=00000000 (sabit)

Fonksiyon : Sayfanın hangi bölümünün hangi hassasiyet ile okunacağının belirtilmesinde kullanılır. Bu fonksiyon her tarama işleminden önce mutlaka kullanılmalıdır.

Çıkış : AX>=0 : Normal.

AX=-1 : Kart takılı değil.

3- Bir satır görüntü bilgisinin okunması:

Giriş : AH=2 (fonksiyon kodu)

ES:DI= RUN-LENGTH tipinde , görüntünün saklanacağı bellek adresi.

Fonksiyon : Görüntü tarayıcıdan bir satır (bu yazı satırı ile karıştırılmamalıdır) görüntü okur , bunu RUN-LENGTH tipinde bilgiye dönüştürür ve belirlenen bellek adresine yazar. Run-length bilgisi 127 veya daha az ise bir byte, daha fazla ise üç byte kullanılır. Bir byte'lık kullanımda pozitif sayılar ardışıl beyaz nokta sayısını , negatif sayılar ardışıl siyah nokta sayısını belirtir. Üç byte'lık kullanımda ilk byte 0 içerir , sonraki iki byte Run-length bilgi içerir (pozitif değerler beyaz , negatif değerler siyah olmak üzere).

Çıkış : AX>=0 : Normal (belleğe yüklenen run-length tipinde görüntünün byte olarak miktarı)

AX=-1 : Kart takılı değil.

AX=-2 : Ünite TIMEOUT. Belirlenen süre içinde üniteye kağıt takılmadı.

AX=-3 : Sayfa sonuna gelindi. Bir sayfa daha önce okunmuş ve son bilgi yollanmıştı.

AX=-4 : Ünite aşırı ısındı. IX-8 fluorescent tüpünün ısınmış olması nedeniyle üç dakika kadar bekleyin.

AX=-5 : Sayfa uzunluğu fazla. Ünitedeki sayfa uzunluğu A4 boyutlarından fazla. IX-8 bu durumda aşırı ısınmaya karşı kendini kapatarak korumaya alır.

CX=görüntü satırındaki toplam nokta sayısı.

4- Bir satır görüntünün okunmadan atlanması:

Giriş : AH=3 (fonksiyon kodu)

Fonksiyon : Bir görüntü satırının okunmadan atlanmasını sağlar.

Çıkış : madde 3 de anlatıldığı gibi.

5- Run-length tipindeki görüntünün nokta görüntü haline genişletilmesi:

Giriş : AH=4 (fonksiyon kodu)

AL=XYY000ZZ (Genişletme modu)

X:Bilgi modu:

0:Normal. Run-length tipindeki datada beyaz noktalar 1 , siyah noktalar 0 ile belirtilmiştir.

1:İnvers. Run-length tipindeki datada beyaz noktalar 0 , siyah noktalar 1 ile belirtilmiştir.

YY:Yazma modu:

00:Eski görüntünün üstüne yazmak.

01:Eski görüntü ile AND işlemi yaparak yazmak.

10:Eski görüntü ile OR işlemi yaparak yazmak.

11:Eski görüntü ile XOR işlemi yaparak yazmak.

ZZ:Renk düzeltme modu:Ölçeklendirme sırasında bazı noktalar kağıdın zemin rengine göre yeniden renklendirilmelidir.

00:Düzeltme yapılmayacak.

01:Beyaza düzeltme yapılacak.

10:Siyaha düzeltme yapılacak.

DS:SI= RUN-LENGTH tipinde , görüntünün bulunduğu bellek adresi.

ES:SI=Nokta Görüntünün yazılacağı bellek adresi.

CX=Byte cinsinden Run-length görüntü uzunluğu.

BX=Ölçeklendirme faktörü.

0: Aynı

-1: İki misli

-2: Üç misli

-3: Dört misli

1 den 256 ya: 1/64 den 256/64 ölçeklendirme.

DL=Başlangıç bit kayıklığı.(0 dan 7 ye kadar)

Fonksiyon : Üç numaralı fonksiyon ile okunan run-length formundaki görüntü bilgisini nokta görüntü haline genişletir.

Çıkış : AX=belleğe yazılan nokta miktarı.

6- Nokta görüntü tipindeki bilginin run-lenght tipine dönüştürme:

Giriş : AH=5 (fonksiyon kodu)
DS:SI=Nokta görüntü tipindeki dataların adresi.
ES:SI=Run-length tipindeki bilgileri saklayacak bellek adresi.
CX:Nokta sayısı.
BL:Bit başlangıç noktası.(0 dan 7 ye)

Fonksiyon : Nokta görüntü tipindeki bilgileri run-length tipine dönüştürür. Bu görüntünün saklanmasında veya iletilmesinde avantaj sağlayacaktır. Zira görüntü daha az bellek işgal edecektir.

7- Run-length bilginin uzunluğunu değiştirme:

Giriş : AH=6 (fonksiyon kodu)
DS:SI=Run-length tipindeki dataların adresi.
ES:SI=Run-length tipindeki yeni bilgileri saklayacak olan bellek adresi.
CX:Dönüştürülecek bilginin byte olarak uzunluğu.

Fonksiyon : Sekiz veya onaltı bitlik run-length görüntü bilgisinin formatının dört bit olarak değiştirilmesi. Bu işlem görüntüye bağlı olarak bellekten tasarruf sağlayabilir aksi takdirde hiç bir işe yaramaz. Görüntü daha çok dikey çizgilerden oluşuyorsa daha etkin bellek kullanımı sağlayacaktır.

Çıkış : AX=belleğe yazılan run-lenght byte miktarı.

8- Dört bit Run-length bilginin uzunluğunu değiştirme:

Giriş : AH=7 (fonksiyon kodu)

DS:SI=Run-length tipindeki dataların adresi.

ES:SI=Run-length tipindeki yeni bilgileri saklayacak olan bellek adresi.

CX:Satır başı bilgisi.

Fonksiyon : Dört bitlik formattaki görüntü bilgilerinin orijinal formatına geri dönüştürülmesi.

Çıkış : AX=belleğe yazılan run-lenght byte miktarı.

9- Nokta görüntü tipindeki datanın 90 derece döndürülmesi:

Giriş : AH=8 (fonksiyon kodu)

AL=Döndürme yönü:

0:Saat yönünde.

1:Saatin ters yönünde.

CX=Döndürülecek görüntünün byte olarak genişliği.

DX:Döndürülecek görüntünün toplam genişliği.

DS:SI=Run-length tipindeki dataların adresi.

Fonksiyon : Görüntünün 90 derece döndürülmesi.

Görüntü Tarayıcıdan Bir Sayfanın Okunarak

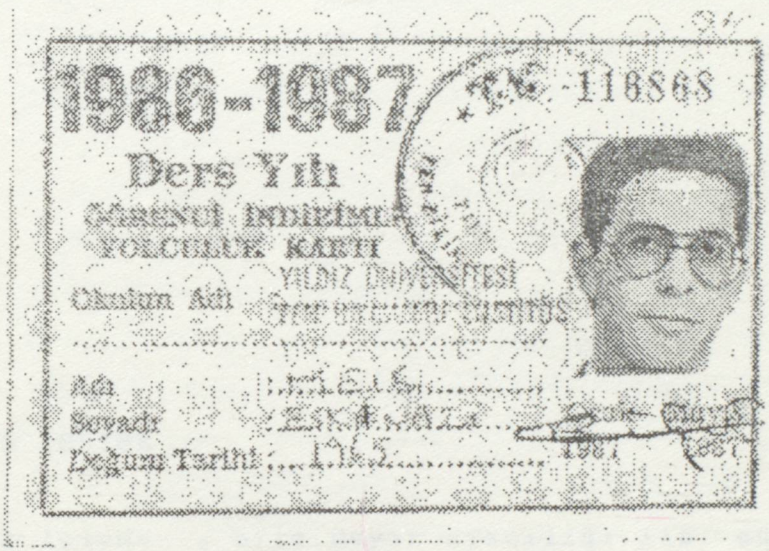
Ekrana Getirilmesi:

```
          PAGE      66,80
TITLE     GÖRÜNTÜ   OKUMA
;
STACKSG   SEGMENT   PARA STACK 'STACK'
          DW        1024 DUP(?)
STACKSG   ENDS
;
DATASG    SEGMENT   PARA 'DATA'
A         DW        1024 DUP(?)
DATASG    ENDS
;
CODESG    SEGMENT   PARA 'CODE'
MAIN      PROC      FAR
          ASSUME    CS:CODESG,DS:DATASG,SS:STACKSG,ES:DATASG
START:
; geri dönüş parametrelerinin düzenlenmesi
          PUSH      DS
          SUB       AX,AX
          PUSH      AX
          MOV       AX,DATASG
          MOV       DS,AX
; ekran modunun seçilmesi
          MOV       AL,6
          MOV       AH,0
          INT       10H
; arabirimin konrol edilmesi
```

```
        MOV     AH,0
        INT     7FH
        CMP     AX,0
        JL      EXIT
; görüntü okuma parametrelerinin verilmesi
        MOV     AH,1
        MOV     AL,020H
        MOV     BX,1
        MOV     CX,1000
        MOV     DX,0
        INT     7FH
        CMP     AX,0
        JL      EXIT
; görüntü orijininin seçilmesi
        MOV     DI,0
; döngü sayacının set edilmesi
        MOV     CX,90
; döngü sayacının saklanması
DON:    PUSH    CX
; bir görüntü satırının okunması
        MOV     AX,DS
        MOV     ES,AX
        PUSH    DI
        MOV     DI,0
        MOV     AH,2
        INT     7FH
        CMP     AX,0
        JL      EXIT
; Run-Length'in nokta nokta görüntüye
; dönüştürülmesi
        MOV     CX,AX
        MOV     SI,0
        POP     DI
```

```
        MOV     AX,0B800H
        MOV     ES,AX
        MOV     BX,32
        MOV     DX,0
        MOV     AH,4
        MOV     AL,0
        INT     7FH
; bir görüntü satırının okunması
        MOV     AX,DS
        MOV     ES,AX
        PUSH    DI
        MOV     DI,0
        MOV     AH,2
        INT     7FH
        CMP     AX,0
        JL      EXIT
; Run-Length'in nokta nokta görüntüye
; dönüştürülmesi
        MOV     CX,AX
        MOV     SI,0
        POP     DI
        ADD     DI,2000H
        MOV     AX,0B800H
        MOV     ES,AX
        MOV     BX,32
        MOV     DX,0
        MOV     AH,4
        MOV     AL,0
        INT     7FH
; döngü  içinde arka arkaya iki okuma ve
; iki dönüşüm bloğunun bulunması ekranın
; interlaced olmasından dolayıdır.
;
```

```
; ekrandaki yeni orijinin tesbit edilmesi
          SUB       DI,2000H
          ADD       DI,80
; döngü kontrol sayacının geri çekilmesi
          POP       CX
; döngü kontrolü
          LOOP      DON
; ana programa (veya DOS'a) geri dönüş
EXIT:
; sayfanın geri kalanının görüntü okuyucudan
; dışarı atılması
          MOV       CX,10000
          MOV       AH,9
          INT       7FH
          RET
MAIN      ENDP
CODESG    ENDS
          END       START
```



GÖRÜNTÜ TARAYICI İLE OKUNMUŞ BİR İ.E.T.T. PASOSU

2

GÖRÜNTÜ İŞLEME

Görüntü İşleme , bilgisayar aracılığı ile sürekli tondaki bir resmin dijital hale getirilmesinden sonra bunun üzerinde değişiklikler yapılmasıdır. Bir görüntü işleme sistemi üç temel fonksiyonu içermelidir. Bunlar resmin dijital hale getirilmesi , bunun görüntülenmesi ve görüntü datasının işleme tabi tutulmasıdır. Bunlara ek olarak sistem iki fonksiyon daha içerebilir. Bunlar görüntü datasının sonradan kullanılması amacıyla saklanması ve bir başka ortama iletilebilmesidir. Böyle bir görüntü işleme sistemi yaklaşık 100.000 Amerikan doları civarındadır.

İlk olarak bu sistemin temel elemanı olan bilgisayarı inceleyelim. Yaygın olarak kullanılan IBM PC veya uyumlu bilgisayarlar genelde bu iş için bazı avantaj ve dezavantajlara sahiptirler. IBM PC leri incelediğimizde Enhanced Graphics Adapter'a (EGA) sahip bir sistem bile

maksimum görüntü duyarlılığı olarak 640 kolona 350 satır ve 6 bitlik pixel kullanabilir. Bu da 640*350 lik bir ekranda 64 sabit renk anlamına gelir. IBM PC lere bellek eklemek internal bus'a konulabilecek bir kart aracılığı ile kolaylıkla mümkün olabilmesine rağmen PC AT ler bile MS-DOS yüzünden 640K bytedan daha fazla belleğe direkt olarak erişemezler. Çeşitli software'ler bu sınırın aşılmasında yardımcı olabilmektedirler ama bu görüntü işleme konusunda dayanılmaz bir performans düşüklüğüne sebep olmaktadır. Üstelik hiç bir standarta uymamaktadırlar. Ayrıca IBM PC'lerde mikro işlemcinin doğal yapısından kaynaklanan 64,535 bytelık bir sınır vardır ki, bu da program geliştirme sırasında problem çıkartmaktadır. Fakat bu Intel 80386 temelli bilgisayarlar için geçerli değildir. Öte yandan, Amiga 1000 veya Amiga 2000 incelenildiğinde farklı bir durum ortaya çıkmaktadır. Amiga incelenildiğinde 640 kolona 400 satırlık gibi bir ekran yoğunluğu ortaya çıkmaktadır ki, bu IBM den yüzde onbeş oranında daha yüksek görüntü duyarlılığı demektir. Fakat Amiga'da bir pixel için dört bit kullanılmaktadır ki, bu maksimum 16 renke karşılık düşmektedir. Buna karşılık bu 16 renk 4096 değişik renklik bir paletten seçilebilir. Bu da, 16 değişik gri renk seviyesi anlamına gelir. Böylece IBM PC'lerdeki sabit 64 renk kullanılarak oluşturulabilecek siyah beyaz görüntüden daha sürekli bir görüntü elde edilebileceğini görürüz. Ayrıca Amiga yapısı ve mikro işlemcisi (Motorola 68000) nedeniyle büyük ölçüdeki belleğe kolaylıkla erişebilir. Bellek haritası IBM PC lerde olduğu gibi segment'lerden oluşmaz tersine sürekli ve bir tek alandır. Bu görüntünün tek bir parça halinde bellekte saklanabilmesine ve bir program aracılığıyla kolayca erişilebilinmesine imkan tanır. Amiga sistemlerine bellek miktarının yükseltilmesi IBM deki gibi basit bir işlemdir. Ayrıca yüksek seviyeli

dillerden, büyük diziler tanımlamak ve kullanmak mümkündür.

Örneğin Amiga Lattice C de :

```
unsigned char *image;
image = AllocMem(128000,0);
```

Bu bize 128,000 bytelık bir dizi tanımlar. Ayrıca Amiga fiyat / performans oranı göz önüne alındığında IBM den daha üstündür.

Görüntü işleyebilmek için önce dijital bir görüntü elde etmek gerekir ki bu ancak çeşitli transducer'lar aracılığıyla yapılabilir. Bunlar arasında scanner'lar veya video kameraları en yaygın olanlarıdır. Bu üniteler kendi özel arabirimleri veya standart paralel/seri portlar yardımıyla bilgisayarlara bağlanabilir.

Okunmuş bir görüntüyü bilgisayar aracılığı ile işlemek ancak programlama ile mümkündür. Genel görüntü işleme dilleri olan 'C' , Fortran 77 veya Lisp ile birlikte , özellikle Amiga düşünüldüğünde STRUCTURED Amigabasic de kullanmak mümkündür. Bu bize ekran düzenleme ve mouse kullanımında etkin kolaylıklar sağlayacaktır. Özellikle aşırı hız gerekmiyorsa da diğer yüksek seviyeli dillerden yapı olarak bir fark oluşturmayacaktır. Görüntü okunduktan son kontrast ayarlama , parlaklığı değiştirme ve kenar ortaya çıkartma gibi temel görüntü işlemlerine geçilebilinir.

GÖRÜNTÜ İŞLEME ALGORİTMALARI

Görüntü okuyucu üniteler bir görüntüyü okuyup belleğe yerleştirirler. Bu fotoğraf benzeri sürekli tondaki şekillerin taranması ile pixel olarak adlandırılan dijital parlaklık değerlerinin belirlenmesidir. Çoğunlukla görüntü okuma sistemleri ANALOG/DİJİTAL dönüştürücüye bağlı video kameralarından oluşurlar ki bu sistem video sinyallerini pixellerden oluşan görüntü belleğine yazar. Bu bilgisayarın merkezi işlem birimi tarafından okuma ve yazma amacıyla erişilebildiği ve görüntüleme ünitesinin de sadece okuyabileceği bir bellek olmalıdır. Eğer bir bellek bölgesi görüntü datası içeriyorsa bu FRAME BUFFER olarak adlandırılır. Bilgisayar buradaki pixelleri işlemden geçirdikten sonra görüntüleme birimi DİJİTAL/ANALOG dönüştürücüsü aracılığıyla monitöre gerekli analog sinyalleri sağlar.

Görüntü işleme algoritmalarının sınıflandırılması:

POINT PROCESS: Bu algoritmada bir pixelin değeri sadece ve sadece o pixelin eski değerine bakılarak tayin edilir.

AREA PROCESS: Bu tür algoritmalarda pixelin değerine karar verilirken kendi orijnal değerinin yanı sıra komşularınınki de hesaba katılır.

GEOMETRIC PROCESS: Herhangi bir pixelin koordinatları veya düzeni değişiyorsa bu tür algoritmalar bu sınıfa girer.

FRAME PROCESS: Bu tür algoritmalar birden fazla dijital görüntünün aynı adresli pixelleri üzerinde işlem yaparak o pixelin yeni değerini hesaplar.

Görüntü işlemenin amaçları olarak bir görüntünün iyileştirilmesini, görüntü elemanlarının sınıflandırılmasını ve tanınmasını sayabiliriz. Örneğin uydulardan alınan yeryüzü fotoğraflarının işlenerek çeşitli şartlara uyan bölgelerin belirlenmesi gibi.

POINT PROCESS:

Bu algoritma türünde şeklin her noktasındaki pixellerin yeni değeri sadece kendi değeri ve bazen de kendi adresi aracılığıyla hesaplanır. Örneğin her pixelin parlaklığı yüzde 25 arttırılabilir veya bu parlaklık değeri 40 ton kadar koyultulabilinir. Bu işlem bir resmin tamamının parlaklığının arttırılmasına karşılık düşer. Bu hesap yapılırken fonksiyona pixelin adresi de parametre olarak katılırsa bu kez yapılan iş gölgelemenin değiştirilmesine karşılık düşer. Örneğin bir resmin ortasında 'AURA' yaratmak için kullanılması gereken point process fonksiyonu.

```
output_value = input_value*k*exp(-(x*x/l+y*y/l))-m
```

Bu fonksiyon şeklin ortasını daha parlak kenarlarını ise daha koyu hale getirecektir. Point process algoritmalarına yardımcı olarak INTENSITY HISTOGRAM ları oluşturulabilinir. Bu şeklin renk tonları ile ilgili istatistiksel bir diagramdır ve point process için gerekli parametrelerin hesaplanmasında kullanılır. Bu diagramın oluşturulması için muhtemel her pixel değerine karşılık düşecek şekilde bir dizi tanımlanır. Eğer sekiz bitlik pixel değeri kullanıyorsak bu 256 muhtemel değişik pixel tonu anlamına gelecektir ki 256 elemanlı bir dizi bu iş için yeterli

olacaktır. Her pixel tek tek incelenerek kendi renk tonuna karşılık düşen dizi elemanı bir arttırılır. Bu, şekil ile ilgili istatistiksel ölçümleri içeren bir hesap türü olmakla beraber her adımda bir tek nokta ile ilgilenildiğinden point process sınıfına girer (her ne kadar görüntü üzerinde bir değişiklik yapmıyorsa bile). Bu histogram bilgileri görüntünün kontrastını değiştirmede yararlı olacaktır. Belli bir eşik değeri tesbit edildikten sonra 0 nolu histogram değerinden başlayarak bu eşik değerini ilk aşan histogram noktası tesbit edilir. Bu işlem 255 den geriye doğru bir kere daha tekrarlanınca elde edilen bu iki değer point process algoritmasında parametre olarak kullanılabilinir. Bu iki değer gerçekte şekildeki en çok pixel miktarını içeren en dar banttır. Bu işlem görüntüdeki bazi bilgilerin kaybolmasına neden olabilir. Bu iki histogram değerinin arasında kalan detaylar ortaya çıkarken bunun dışında kalanlar ise kaybolur. Böylece eşik değerlerinin dikkatli ve her şekle göre farklı tesbit edilmesi zorunluluğu ortaya çıkar. Ayrıca hangi algoritmaların ne sırayla uygulanacağı da ayrı bir problemdir. Zira bazı algoritmalar detayları yok etmekte bazıları ise abartmaktadır. Point processin örneklerinden biri de PSEUDOCOLORING dir. Bu gri tonlardan oluşan bir şekilde belli bir değerdeki veya aralıktaki pixellerin gri tonları dışındaki renklerle ifade edilmesidir. Örneğin pixel değeri 150 ila 200 arasında olan bölgelerin belirlenmesi ve bunların kırmızı olarak görüntülenmesi gibi. Geri kalan bölgeler ise değiştirilmeden orijinal gri tonlarıyla ifade edilir. Fakat point procces ile en az 256*256 pixelin işleneceği düşünülürse bu her pixel başına en az iki bellek erişimi , bir çarpma , bir toplama anlamına gelir. Bu tür işlemler için LOOKUP TABLE kullanmak mümkündür zira aynı değere sahip yüzlerce

noktanın yeni değerlerini ayrı ayrı tekrar tekrar hesaplamanın hiç bir gereği yoktur. Bu nedenle her pixel değerine karşılık düşecek yeni değerler hesaplanarak bir tabloya yerleştirilir. Böylece her pixelin yeni değeri, bu tablodan eski değeri indis olarak kullanılması aracılığıyla bulunur. Ayrıca bazı bilgisayarların görüntüleme üniteleri bu tablolara doğrudan erişebilmek suretiyle herhangi bir başka işleme gerek duyulmadan istenileni yapabilmektedir (LUT).

AREA PROCESS:

Bu process türü pixellerin yeni değerlerini tayin etmek için kendisinin ve komşularının değerlerini kullanır. Area process daha çok istenmeyen veya tekrarlanan pixellerin filtrelenmesinde kullanılır. Bu filtreleme türü istenmeyen gürültülerin (analog şeklin dijital hale dönüştürülmesi sırasında oluşan algılama hataları) yok edilmesinde , şekildeki detayların ortaya çıkartılmasında veya renk tonu geçişlerinin yumuşatılmasında kullanılabilir. Bu yönten temel olarak bir pixelin değerini katsayılar ile çarptığı komşu pixellerin değerlerinin toplanmasında ibarettir. Bu katsayılara CONVOLUTION KERNEL denir fakat biz kısaca KERNEL olarak adlandıracağız. Örnek olarak her pixelin en yakın sekiz komşusunun ve doğal olarak kendisinin de bulunacağı yani 3*3 lük bir pixel matrisi düşünelim. k(m,n) Kernel değeri , p(m,n) pixel değeri olmak üzere üzere formülümüz aşagıdaki şekilde ifade edilir.

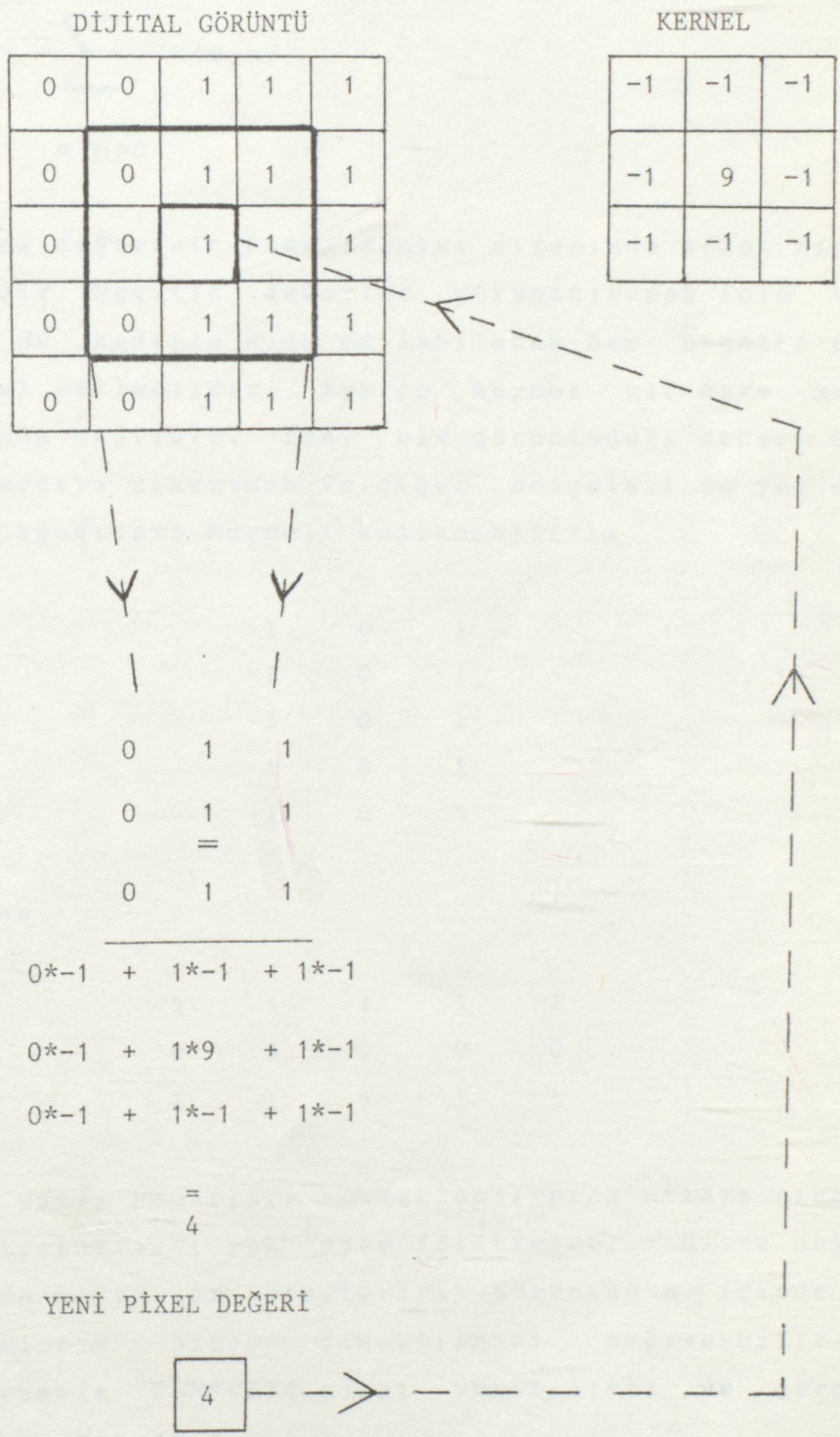
$$p(x,y) = \sum_{m,n=0} k(m,n) * p(x+m-1, y+n-1)$$

Bunu açarsak ve örnek olarak (1,1) koordinatlarındaki bir pixeli düşünürsek.

$$p(1,1) = p(0,0)*k(0,0) + p(1,0)*k(1,0) + p(2,0)*k(2,0) + p(0,1)*k(0,1) + p(1,1)*k(1,1) + p(2,1)*k(2,1) + p(0,2)*k(0,2) + p(1,2)*k(1,2) + p(2,2)*k(2,2)$$

formülünü elde ederiz. Tüm şeklin filtrelenmesi için her pixele bu işlemin tek tek uygulanması gerekmektedir. Bunu küçük kernel matrisimizin görüntü matrisi üzerinde kayarak her noktadan geçtiği gibi düşünebiliriz. Her noktaya formül uygulanarak sonuç kernel matrisinin ortasına rastlayan pixelin yeni değeri olarak ortaya çıkar. Fakat bu 3*3 lük bir kernel, 256 ya 256 lık bir görüntüye uygulanacağı zaman 3*3*256*256 adet yani 589,824 toplam ve çarpma işlemi anlamına gelir. Bu büyük işlemin maliyeti ya çok uzun zaman yada çok fazla para olarak ortaya çıkacaktır.

Kernel matrisi tüm görüntü datasının üzerinde dolaştırıldığında bu matrisin büyüklüğüne bağlı olarak kenarda kalan bazı bölgelerin yeni değerleri hesaplanamayacaktır. Örneğin 3*3 lük kernel kullanıldığında tüm görüntünün kenarlarında bir pixel genişliğinde bir bantın yeni değeri hesaplanamayacaktır. Bu bantın genişliği kernel boyutunun yarısı kadardır. Bu çerceve kısmı ihmal edilebileceği gibi belirli bir değere çekilir yada anlamlı değer taşıyan en yakın komşusunun değeri kopyalanır. Kernel matrisindeki değerlere bağlı olarak bu matrisi ile işlem gören pixelin yeni değeri kendisine ayrılmış olan bitlere sığmayabilir. Bu durumun bilgi kaybına sebep olmaması için elde edilen her yeni değerin sabit bir sayıya bölünmesi gerekebilir. Bu sayı genelde tüm kernel çarpım katsayılarının toplamına eşittir.


DİJİTAL GÖRÜNTÜ
KERNEL
0 0 1 1 1
0 0 1 1 1
0 0 1 1 1
0 0 1 1 1
0 0 1 1 1
-1 -1 -1
-1 9 -1
-1 -1 -1
0 1 1
0 1 1
=
0 1 1
0*-1 + 1*-1 + 1*-1
0*-1 + 1*9 + 1*-1
0*-1 + 1*-1 + 1*-1
=
4
YENİ PİXEL DEĞERİ
4

$$k = \sum_{m,n=0} k(m,n)$$

Pixelin yeni değerinin hesaplanması sırasında sonuç negatif de çıkabilir. Negatif değerler görüntülenmek için uygun değildir. Bu nedenle elde edilebilecek her negatif sonuç sıfır kabul edilmelidir. Ayrıca kernel bir kare matris olmak zorunda değildir. Eğer bir görüntüdeki sadece dikey kenarları ortaya çıkartmak ve diğer bölgeleri de yok etmek istiyorsak aşağıdaki kerneli kullanabiliriz.

-1	0	1
-1	0	1
-1	0	1
-1	0	1
-1	0	1

Aynı şekilde,

-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1

Kerneli de yatay kenarları kuvvetlendirerek ortaya çıkartır ve diğer ayrıntıları yok eder (filtreler). Hatta değişik patternlerde kerneller oluşturarak görüntünün içinde buna uyan cisimlerin ortaya çıkartılması sağlanabilir. Bu durumda kernela TEMPLATE adını verebiliriz ve gerçekte korolasyon yapmış oluruz.

Bir görüntüde frekanstan bahsedildiğinde aynı tondaki pixellerin arka arkaya tekrarlanış yoğunluğu kastedilmek istenir. Buna göre bir görüntü için yüksek frekans filtresi ani parlaklik değişimlerini tesbit etmelidir. Bunun için kullanmamız gereken kernel:

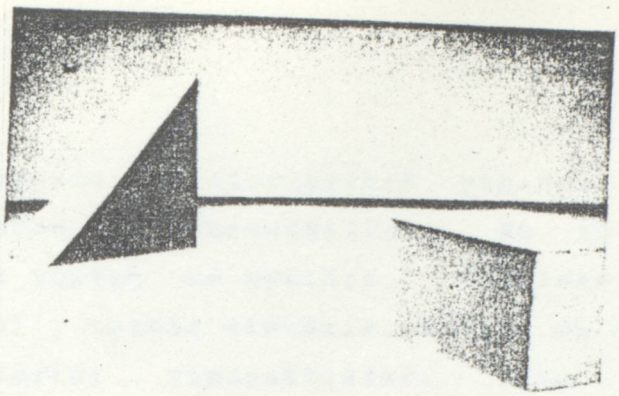
-1	-1	-1
-1	8	-1
-1	-1	-1

Bu kernel genelde LAPLACIAN filtresi olarak bilinir. Bu kernelin uygulandığı görüntülerde yüksek frekansa sahip kenarlar (ani parlaklik değişimleri) ortaya çıkacak diğer alçak frekanslar (ani parlaklık değişiminin olmadığı yüzeyler) ise ortadan kalkacaktır. Bu nedenle yukarıdaki kernel 'KENAR TESBİT EDİCİ' olarak kullanılır.

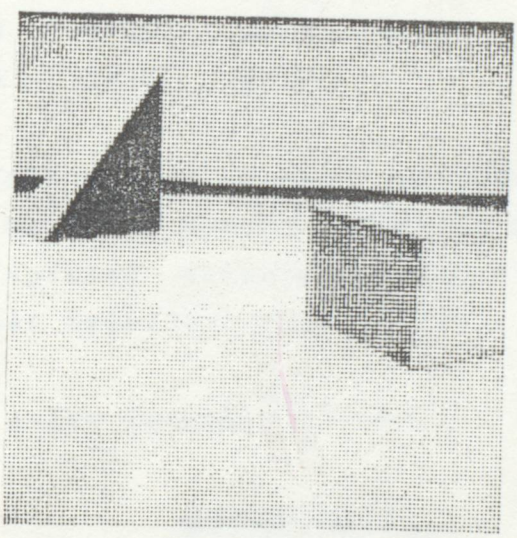
Eğer Laplacian kernelında küçük bir değişiklik yaparak ortadaki pixelin katsayısını 8 yerine 9 yaparsak

-1	-1	-1
-1	9	-1
-1	-1	-1

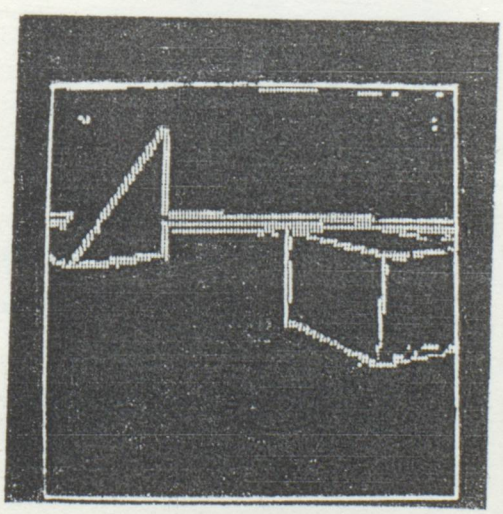
matrisini elde ederiz. Bununla yine aynı sonucu elde ederiz yani ani parlaklık değişikliklerini tesbit ederiz. Fakat bir farkla , ani parlaklık değişikliklerinin olmadığı yani frekansın değişmediği bölgeleri orijinal değerinde bırakır. Bu kernel frekansın yüksek olduğu bölgeleri seçerek bunları ortaya çıkartır yani abartır. Sonuçta oluşan yeni görüntü daha keskin hatlara sahip olur. Fakat bu yüzden de şekildeki gürültüler (algılama hatalarından doğan yanlış pixel değerleri) artar.



ORİJİNAL GÖRÜNTÜ



DİJİTAL ŞEKİL



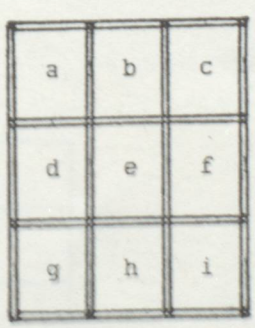
LAPLACIAN FİLTRESİ

Yüksek frekans filtrelerinin yanında alçak frekans filtrelerinden de bahsedebiliriz. Bu tür filtreler için uygulanacak yöntem de aynıdır. Yapılması gereken sadece doğru kernel'ı tesbit etmektir. Fakat bu tür filtreler ton değişikliklerini yumuşattıkları için orijinal şekli bulanıklaştırırlar. Böyle olunca da bu kez gürültüler kaybolur yada küçülür. Alçak ferkans filtresi olarak kullanılabilecek kernel'lar

1/10	1/10	1/10		0.1	0.1	0.1
1/10	2/10	1/10	veya	0.1	0.2	0.1
1/10	1/10	1/10		0.1	0.1	0.1

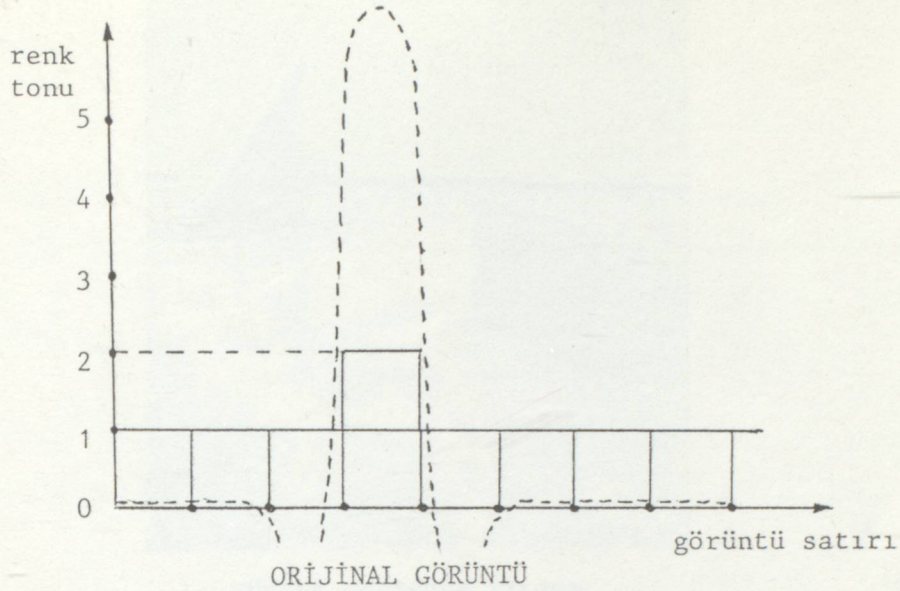
olabilir. Yine burada dikkat edilmesi gereken bir konu vardır. Bu da hesap sonucu oluşacak yeni pixel değerinin onun için ayrılan bitlere taşmadan sığabilmesidir. Bu nedenle kernel değerleri yine dikkatli seçilmelidir.

Değişik bir alçak frekans filtresini incelersek:



Şeklinde 3*3 lük bir matris göz önüne alındığında ortadaki pixelin yeni değeri

e'=((NOT e)&(a&b&c&d&f&g&h&i))!(e&(a!b!c!d!f!g!h!i))

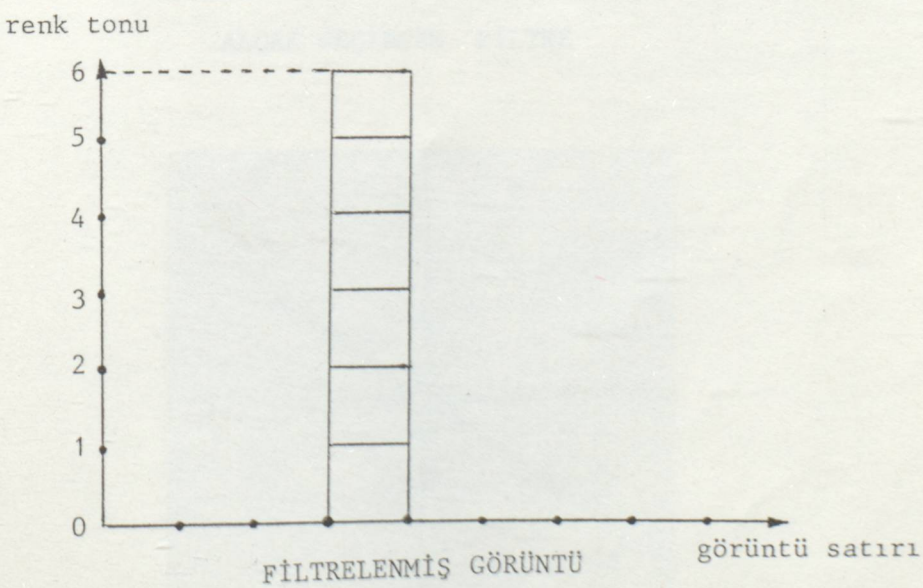


ORİJİNAL GÖRÜNTÜ

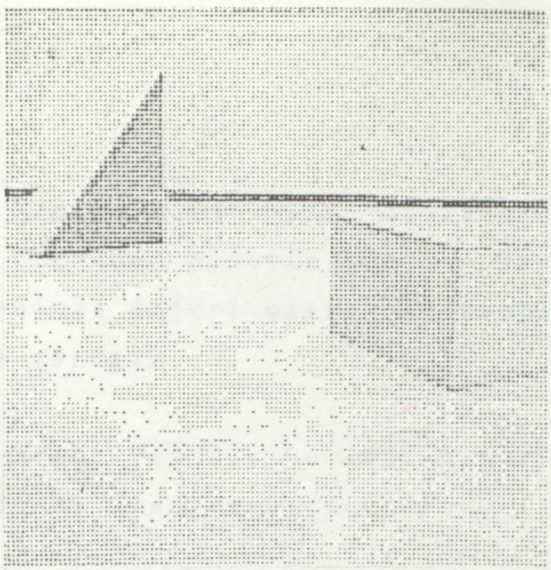
KERNEL

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

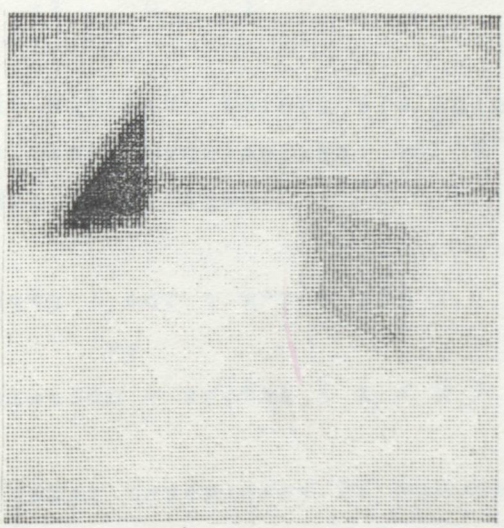
Orijinal görüntü LAPLACIAN KERNEL ile filtrelenmesi,



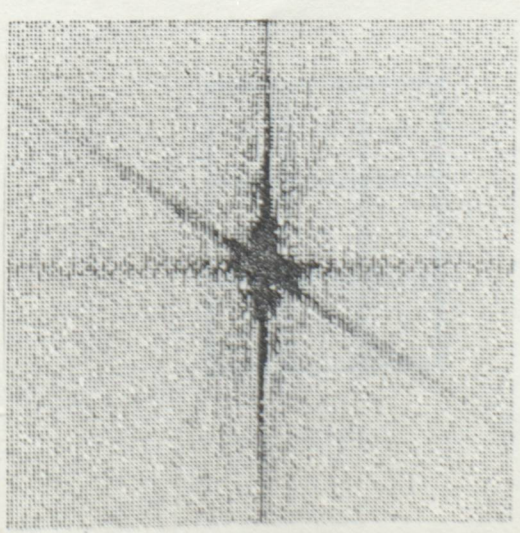
FİLTRELENMİŞ GÖRÜNTÜ



YÜKSEK GEÇİRGEN FİLTRE



ALÇAK GEÇİRGEN FİLTRE



FOURIER TRASFORMU

Şeklinde e pixelinin yeni değerini hesaplayabiliriz. Bu operatör, eğer bir pixelin bütün yakın komşuları aynı renk ise, yeni pixel değeri olarak bu rengi verir, aksi takdirde değişiklik yapmaz.

Nonlinear Area Process Bu tür algoritmalar daha iyi bir sinyal-gürültü oranı sağlar.

$$X: \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad Y: \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

Şeklinde iki değişik kernel ele alınır. Her pixele bu iki matris tek tek uygulanırsa ve X , Y değerleri elde edilirse. Her pixel için

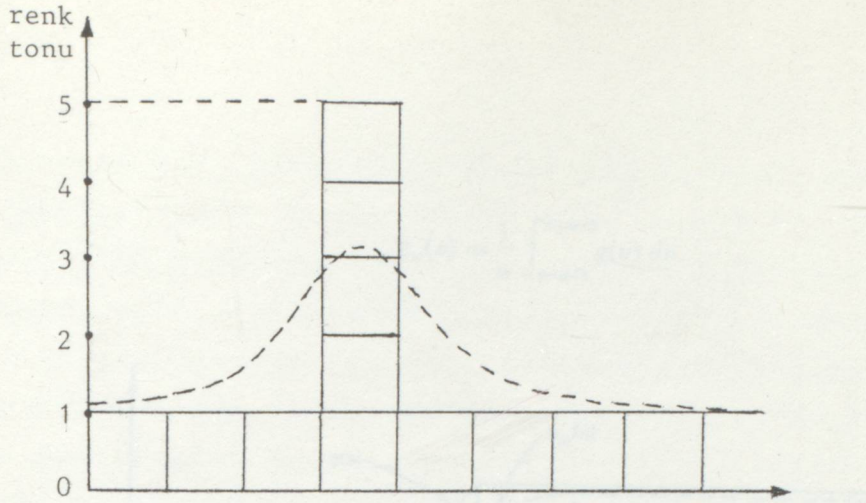
$$strength = sqr (x*x + y*y)$$

ve

$$orientation = arctan (x / y)$$

olarak iki değer hesaplarız ki bununun sonucunu iki boyutlu şekillerle göstermek mümkün değildir. Bu filtreye SOBEL adı verilir ki biz bunun iki boyutlu yüzeylerde anlamlı olan bir başka şeklini inceleyeceğiz. Buna göre

a	b	c
d	e	f
g	h	i

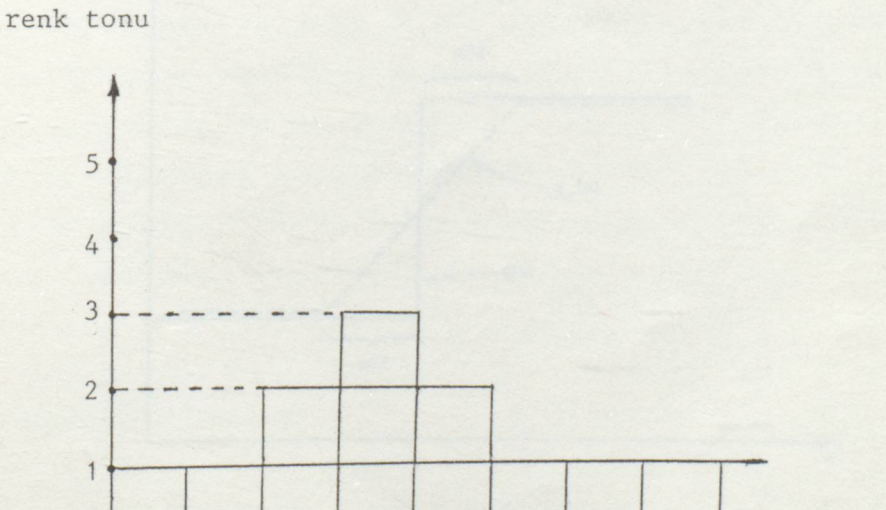


ORİJİNAL GÖRÜNTÜ

KERNEL

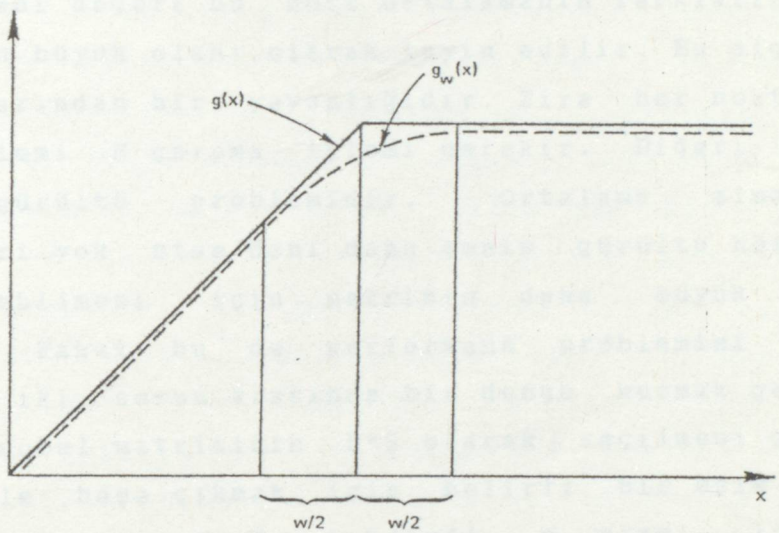
0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

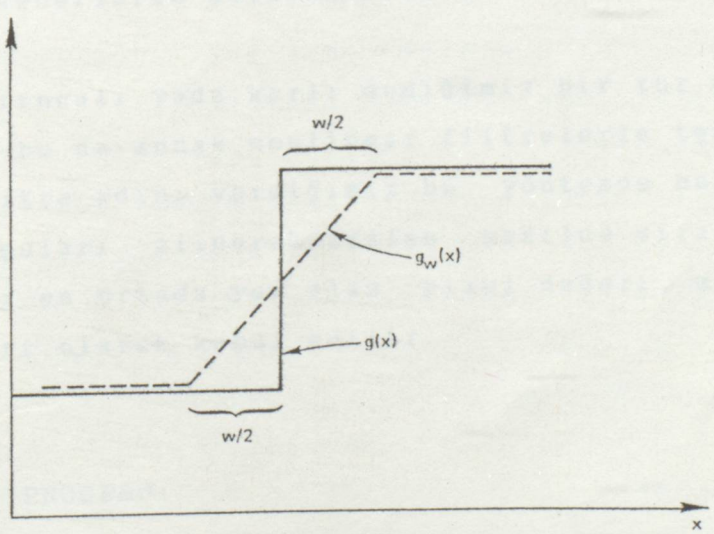
Orijinal görüntü YUMUŞATMA KERNEL'i ile filtrelenmesi,



FİLTRELENMİŞ GÖRÜNTÜ

$$g_w(x) = \frac{1}{w} \int_{x-w/2}^{x+w/2} g(u)\, du.$$





KENAR/KÖŞE YUMUŞATMA ÖRNEKLERİ

Şeklindeki bir matrisin ortasındaki pixelin yeni değerini hesaplamak için a-e-i , b-e-h , c-e-g ve d-e-f pixel gruplarının değerleri ayrı ayrı toplanır ve bu sonuçlar 3'e bölünür yani aritmetik ortalamaları alınır. Ortadaki pixelin yeni değeri bu dört ortalamanın farklarının mutlak değerce en büyük olanı olarak tayin edilir. Bu algoritmanın kötü yanlarından biri yavaşlığıdır. Zira her nokta için 16 toplam işlemi 8 çarpma işlemi gerekir. Diğeri ise doğal olarak gürültü problemidir. Ortalama almak küçük gürültüleri yok etse dahi daha geniş gürültü bölgelerinin yok edilebilmesi için matrisin daha büyük seçilmesi gerekir. Fakat bu da performans problemini arttırır. Kısaca bu iki sorun arasında bir denge kurmak gereklidir. Örneğin sobel matrisinin 5*5 olarak seçilmesi gibi. Hız problemiyle başa çıkmak için belirli bir eşik seviyesi seçilerek bu aşıldığı takdirde o pixel için hesap yapılmasına son verilebilinir. En son hesaplanan değer o pixelin yeni değeri olarak kabul edilir. Fakat her zaman lekelerin kenar zannedilme olasılığı ve birbirine yakın kademeli kenerların kaybedilmesi problemi vardır.

Ayrıca karıncalı yada karlı dediğimiz bir tür görüntü şekli vardır ki bu da ancak nonlinear filtrelerle temizlenebilir. Median filtre adını verdiğimiz bu yöntemde her pixel ve en yakın komşuları alınarak azalan sekilde sıralanırlar. Bu sıralamada en ortada yer alan pixel değeri, merkez pixelin yeni değeri olarak kabul edilir.

GEOMETRIC PROCESS:

Geometric process algoritmaları pixellerin yerlerini ve düzenlerini değiştirmek için kullanılırlar. Bu genelde

bakış açısını değiştirmek, kameralardaki distorsiyonu düzeltmek veya arzu edilen bölgelerin boyutlarını değiştirmek için kullanılır. Ayrıca bazen dökümanlar arzu edildiği yönde taranamayabilir ve bu yüzden tanıma algoritması için doğru yöne çevrilmesi gerekir. Bu tür algoritmalar arasında döndürme, daraltma, genişletme, yer değiştirme sayılabilir. Kısaca (x,y) koordinatlarındaki pixelin yeni (x',y') koordinatlarına taşınmasıdır. Daraltma ve genişletmede bu aktarma sırasında bazı pixellerin adresleri çakışabilir. Örneğin 90 derece döndürmede (x,y) koordinatlarındaki bir pixel ((YSIZE-1)-y,x) koordinatına taşınır. Bir başka yaklaşımla bakılırsa , her yatay görüntü satırının dikey olarak yada arzu edilen herhangi bir açı altında yeni yerine yerleştirilmesidir. Fakat area process de olduğu gibi bütün bu işlemler bir buffer kullanılarak yapılmalıdır. Aksi takdirde her yeni işlem bir öncekilerden etkilenecek ve istenilen işlem sağlanamayacaktır. Öte yandan bu buffer'ının uzunluğu bilgisayar sisteminin hardware yapısına bağlı olarak bazı limitleri aşamayabilir.

Görüntü okuyucunun yapısına bağlı olarak bilgiler değişik standartlarda olabilir yada görüntü saklamak veya transfer etmek için değişik çözümleme ve kod dönüştürme algoritmalarıda bu sınıfa girer. Görüntüleme üniteleri tarafından kullanılan formata RASTER adı verilir. Görüntü okuma ünitelerinin kullandığı format ise genelde CCITT (International Telegraph and Telephone Consultative Committee) in standardına uyar ki bu faksimile data yoğunlaştırma formatıdır. Bu, görüntü okuma ünitelerinin faksimile karakteristiği göstermeleri yüzünden böyledir. Biz bu algoritmaları üç temel sınıfta inceleyeceğiz. Bunlar,

Görüntü formatının (yoğunluğunun) değiştirilmesi:

- Decoder (çözümleyici)
- Encoder (yoğunlaştırıcı)

Ölçeklendirmenin değiştirilmesi:

- Sabit oranda büyültme-küçültme
- Genel büyültme-küçültme

Döndürme

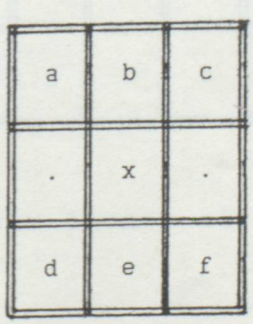
- 180 derece döndürme
- 90 derece döndürme(saat yönünde ve saatin ters yönünde)

Görüntü formatının (yoğunluğunun) değiştirilmesi: Normal (Raster) bir görüntüde ayrıntının bulunmadığı bölgeler için de pixel başına sabit bir miktarda byte harcanır. Bu gereksiz tekrar sadece görüntüleme ünitesi için şarttır. Fakat görüntü saklamak veya transfer etmek için hiçte uygun değildir. Formatı değiştirilerek saklanacak bir şekil daha az bellek gerektirecek ve transfer sırasında daha az zaman harcayacaktır. Bu tür görüntü formatına RUN-LENGHT adını vereceğiz. Raster bir görüntüdeki arka arkaya gelen aynı tondaki veya renkteki her pixel grubuna ise RUN diyeceğiz. Run-lenght adından da anlaşılacağı gibi bu arka arkaya tekrarlanan RUN'ların uzunluklarının hesaplanması sonucu oluşan görüntü formatıdır. Örneğin arka arkaya gelen 20 adet beyaz pixelden sonra 15 siyah pixel ifade edilmek istenildiğinde 20 adet 0 içeren byte ve 15 adet 1 içeren byte yerine, sadece bir byte uzunluğunda 20 ve yine bir byte uzunluğunda 15 (veya -15) toplam iki byte kullanılır. Genelde görüntü okuyucular taradıkları görüntüyü bu formatta bilgisayara aktarırlar. Bu bilgi daha sonra bilgisayar tarafından çözümlenerek görüntülenecek ve işlenecek hale getirilir. Örnek olarak yaklaşık 70 kbyte büyüklüğünde bir sayfa bir milyon komutluk bir iterasyon

aracılığıyla 20 kbyte'a sıkıştırılabilir.

Ölçeklendirmenin değiştirilmesi:

Ölçek değiştirme değişik görüntüleme üniteleri için şeklin boyutlarını değiştirme veya ayrıntıların küçültülmesi-büyütülmesi amacıyla kullanılır. Fakat büyültme sırasında görüntü kalitesi azalır, küçültme sırasında ise bazı gerekli ayrıntılar kaybolabilir. Hardware karakteristiklerin uyuşmaması sonucu özel bazı ölçekleme oranları gerekebilir. Yani görüntü okuyucu ile görüntüleme ünitesinin yoğunluklarının uyuşmaması gibi durumlarda bazı özel ölçekleme oranları istenebilir. Bunun yanında kullanıcılar kendi istekleri doğrultusunda her oranda ölçekleme yapabilirler. Genel ölçekleme yöntemlerinde oran belli olmadığı için oluşacak yeni görüntünün kalitesi hakkında garanti verilemez. Ama özel ölçekleme (yani oranın belli ve sabit olduğu) yöntemlerinde amaca uygun olarak hata minimuma düşürülür. Örneğin 2:1 oranında küçültme, şekle değişik terminaller aracılığıyla izlenebilme imkanı tanımakta yada birkaç değişik resmin aynı anda görüntülenmesi için kullanılır. Örneğin 5:6 büyütme algoritması incelenirse.



şeklinde bir matris göz önüne alındığında x pixelinin yeni değeri,

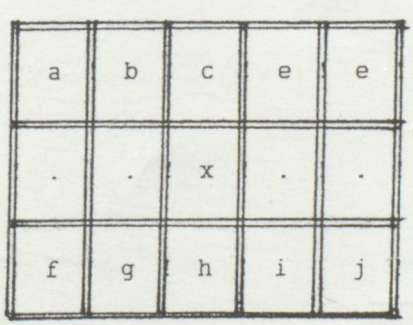
x=(b&e)!((b!e)&((a&f)!(c&d))

veya bilgisayar uygulamasında hesap kolaylığı olsun diye

x=x!((b!e)&((a&f)!(c&d)))

şeklinde basitleştirebiliriz.

Genel ölçeklendirme ise daha değişik bir yöntemle yapılır. Bu yöntemde iki boyutlu tek bir ölçekleme yerine bir boyutta ölçekleme yapıldıktan sonra şekil 90 derece döndürülür ve aynı boyutta (döndürme yüzünden aslında diğer boyutta) bir kez daha ölçekleme yapılır ve daha sonra tersine bir döndürme işlemi yapılır. Yani incelemesi gereken algoritma bir boyutta (örneğin dikey boyutta) ölçekleme yapan olacaktır. Örneğin bu algoritmayı 5:6 oranını örnek kullanarak incelediğimizde;



Şeklinde bir matris göz önüne alındığında x pixelinin yeni değeri,

x=(c!h)&((a&j)!(b&i)!(c&h)!(d&g)!(e&f))

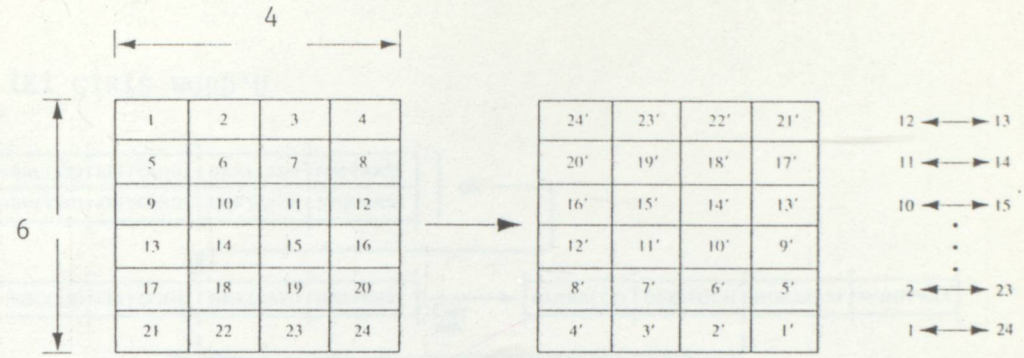
veya bilgisayar uygulamasında hesap kolaylığı olsun diye

x=h!(c&((a&j)!(b&i)!(d&g)!(e&f)))

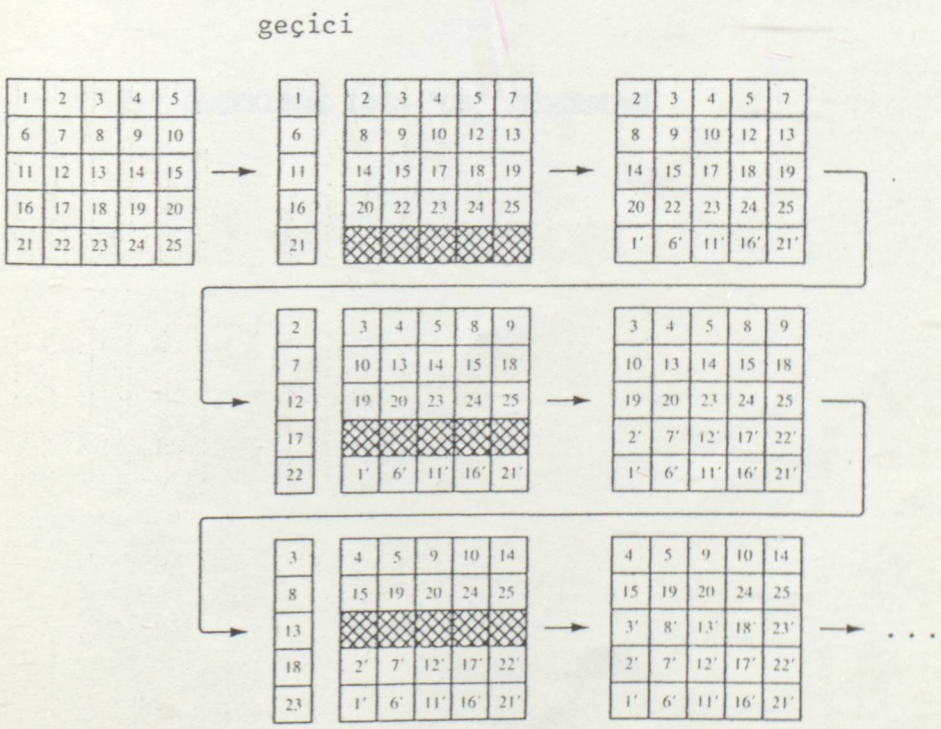
şeklinde basitleştirebiliriz.

Döndürme algoritmaları:

Doğru yönden taranamayan şekillerin istenilen yöne çevirilmesi için 90 derecelik çevirme yetecektir. Ek olarak 180 dercelik döndürmeleri de incelenecektir. Fakat anlaması kolay olduğundan sadece birer örnek yetecektir.

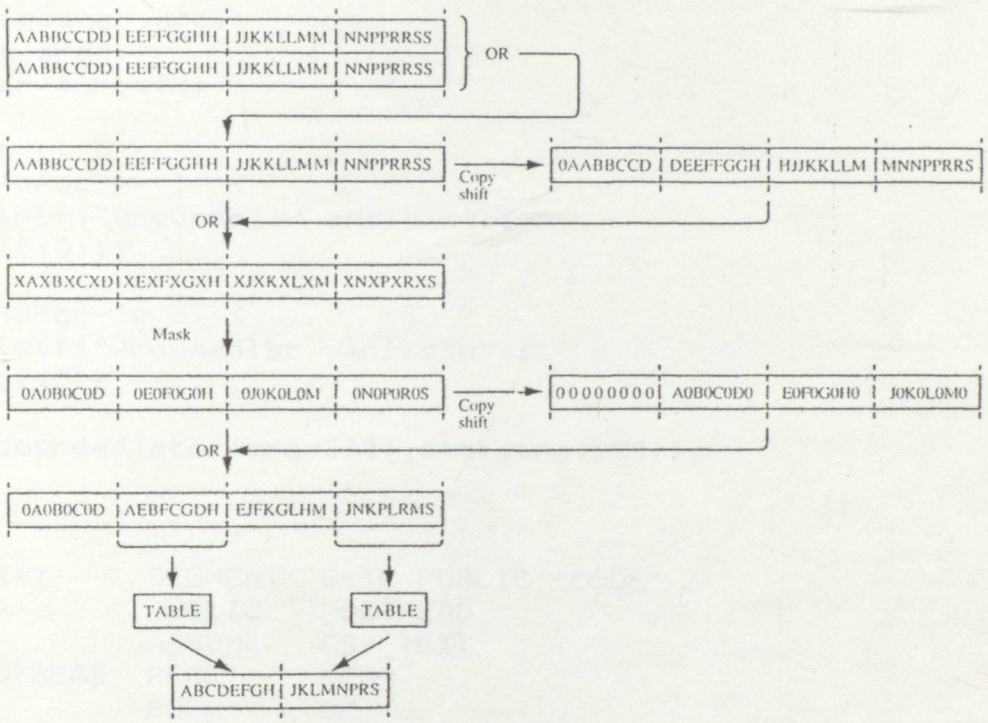


180 DERECE DÖNRÜRME



90 DERECE DÖNDÜRME

İKİ GİRİŞ WORD'Ü



ÇIKIŞ YARIM WORD'U

2:1 ÖLÇEKLEME İÇİN 'OR' PROSEDÜRÜ

PİXEL DEĞERİNİN OKUNMASI

```
#include <stdio.h>
#include <stdlib.h>
main(argc,argv)
int argc;
char *argv[];
{
int c;
if(argc == 1) {
printf("Argumanlar eksik\n");
exit(0);
};
if(argc != 3) {
printf("Argumanlar yanlıs\n");
exit(0);
};
c=dotread(atoi(argv[1]),atoi(argv[2]));
}
```

```
_TEXT       SEGMENT  BYTE PUBLIC 'CODE'
            PUBLIC   _DOTREAD
            ASSUME   CS:_TEXT
_DOTREAD    PROC     NEAR
            PUSH     BP
            PUSH     ES
            PUSH     DS
            PUSH     BX
            PUSH     CX
            PUSH     DX
            MOV      BP,SP
            MOV      DX,[BP+14]
            MOV      CX,[BP+16]
            MOV      AH,13
            INT      10H
            MOV      AX,0
            POP      DX
            POP      CX
            POP      BX
            POP      DS
            POP      ES
            POP      BP
            RET
_DOTREAD    ENDP
_TEXT       ENDS
            END
```

PİXEL DEĞERİNİN DEĞİŞTİRİLMESİ

```
#include <stdio.h>
#include <stdlib.h>
main(argc,argv)
int argc;
char *argv[];
{
int c;
if(argc == 1) {
printf("Argumanlar eksik\n");
exit(0);
};
if(argc != 4) {
printf("Argumanlar yanlıs\n");
exit(0);
};
c=dotset(atoi(argv[1]),atoi(argv[2]),atoi(argv[3]));
}
```

```
_TEXT     SEGMENT  BYTE PUBLIC 'CODE'
          PUBLIC   _DOTSET
          ASSUME   CS:_TEXT
_DOTSET   PROC     NEAR
          PUSH     BP
          PUSH     ES
          PUSH     DS
          PUSH     BX
          PUSH     CX
          PUSH     DX
          MOV      BP,SP
          MOV      DX,[BP+14]
          MOV      CX,[BP+16]
          MOV      AL,[BP+18]
          MOV      AH,12
          INT      10H
          MOV      AX,0
          POP      DX
          POP      CX
          POP      BX
          POP      DS
          POP      ES
          POP      BP
          RET
_DOTSET   ENDP
_TEXT     ENDS
          END
```

Görüntüye Point Process Uygulanması

```
PROGRAM POINTP;
VAR X,Y: INTEGER ;
    SCREEN  : ARRAY (.0..255,0..255.) OF INTEGER;
BEGIN
       FOR Y:=0 TO 255 DO
          FOR X:=0 TO 255 DO
             SCREEN(.X,Y.):=SCREEN(.X,Y.)+15;
END.
```

Görüntü Histogramının Hesaplanması

```
PROGRAM HISTOG;
VAR X,Y: INTEGER ;
    SCREEN  : ARRAY (.0..255,0..255.) OF INTEGER;
    HISTOG  : ARRAY (.0..255.) OF INTEGER;
BEGIN
/* histogram dizisinin boşaltılması */
    FOR X:=0 TO 255 DO HISTOG(.X.):=0;
/* histogramın oluşturulması */
    FOR Y:=0 TO 255 DO
        FOR X:=0 TO 255 DO
        HISTOG(.SCREEN(.X,Y.).):=HISTOG(.SCREEN(.X,Y.).)+1;
END.
```

Görüntü Kontrastının Histogram Yardımı İle Arttırılması

```
 PROGRAM CONTRAST;
 VAR X,Y: INTEGER ;
     SCREEN   : ARRAY (.0..255,0..255.) OF INTEGER;
     HISTOG   : ARRAY (.0..255.) OF INTEGER;
     LOW_BIN,HIGH_BIN : INTEGER;
     STEP,STEP_VALUE  : REAL;
     TRAN     : ARRAY (.0..255.) OF INTEGER;
 BEGIN
/* histogram dizisinin boşaltılması */
   FOR X:=0 TO 255 DO HISTOG(.X.):=0;
/* histogramın oluşturulması */
   FOR Y:=0 TO 255 DO
      FOR X:=0 TO 255 DO
      HISTOG(.SCREEN(.X,Y.).):=HISTOG(.SCREEN(.X,Y.).)+1;
/* histogramın uç noktalarının tesbit edilmesi */
   FOR LOW_BIN:=0 TO 255 DO
                    IF(HISTOG(.LOW_BIN.)>30) THEN LEAVE;
   FOR HIGH_BIN:=255 DOWNTO 0 DO
                    IF(HISTOG(.HIGH_BIN.)>30) THEN LEAVE;
   STEP:=256/(HIGH_BIN-LOW_BIN);
   STEP_VALUE:=0;
/* LUT oluşturulması */
   FOR X:=0 TO LOW_BIN DO TRAN(.X.):=0;
   FOR X:=LOW_BIN TO HIGH_BIN-1 DO BEGIN
      TRAN(.X.):=TRUNC(STEP_VALUE);
      STEP_VALUE:=STEP_VALUE+STEP;
      END;
```

```
    FOR X:=HIGH_BIN+1 TO 255 DO TRAN(.X.):=255;
/* kontrastın arttırılması */
    FOR Y:=0 TO 255 DO
        FOR X:=0 TO 255 DO
            SCREEN(.X,Y.):=TRAN(.SCREEN(.X,Y.).);
 END.
```

Görüntü Ölçeğinin Değiştirilmesi

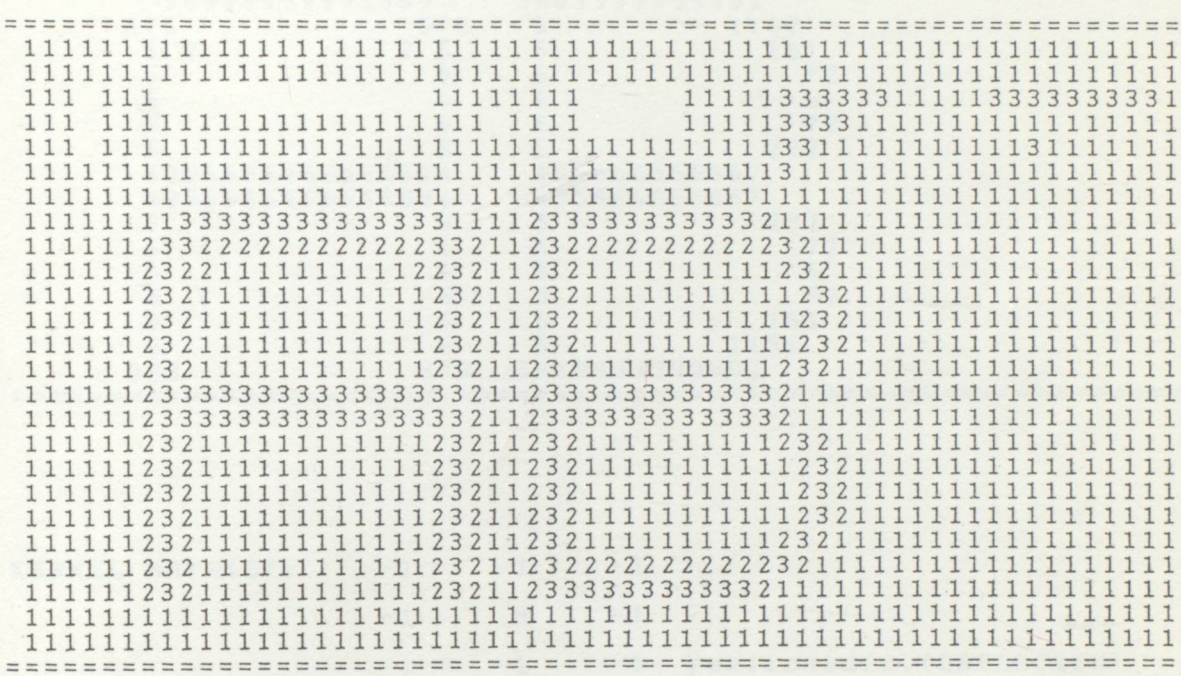
```
PROGRAM SCALE;
VAR I,J,XS,YS,DX,DY,A,B,XA,YA,X,Y: INTEGER ;
    SCREEN  : ARRAY (.0..255,0..255.) OF INTEGER;
    BUFFER  : ARRAY (.0..255,0..255.) OF INTEGER;
BEGIN
/* orijinal görüntünün başlangıç koordinatları */
       XS:=0;
       YS:=0;
/* yeni oluşturulacak olan
görüntünün başlangıç koordinatları */
       X:=0;
       Y:=0;
/* işlem görecek alanın eni ve boyu */
       DX:=255;
       DY:=255;
/* yatay ve dikey ölçek oranları */
       A:=3;
       B:=2;
       FOR I:=0 TO DY DO
          FOR J:=0 TO DX DO
             XA:=XS+ROUND(J/A);
             YA:=YS+ROUND(I/B);
             BUFFER(.X+J,Y+I.):=SCREEN(.XA,YA.);
END.
```

Kernel Matrisi İle Filtreleme Yapılması

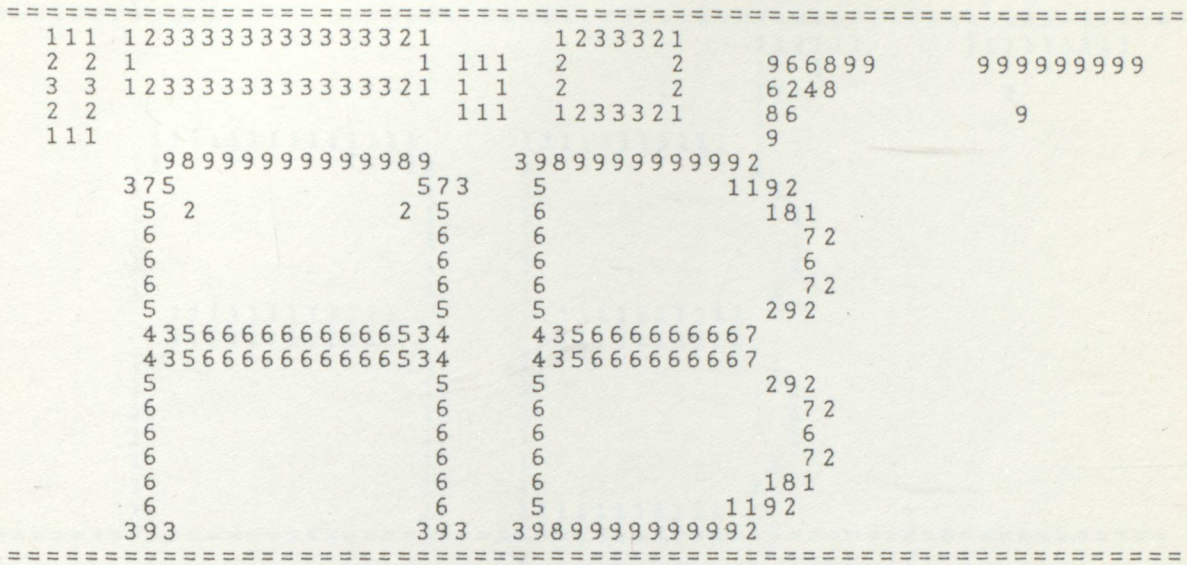
```
PROGRAM KERNEL;
VAR I,J,IA,IE,JA,JE,T: INTEGER;
  SCREEN   : ARRAY (.0..255,0..255.) OF INTEGER;
  BUFFER   : ARRAY (.0..255,0..255.) OF INTEGER;
BEGIN
FOR I:=1 TO 254 DO
  FOR J:=1 TO 254 DO BEGIN
    IA:=I+1;
    IE:=I-1;
    JA:=J+1;
    JE:=J-1;
/* pixellerin kernel matrisi ile çarpılması */
      /* KERNEL :     .1 .1 .1
                      .1 .2 .1
                      .1 .1 .1   */
    T:= +SCREEN(.IE,JE.)+  SCREEN(.IE,J.)+SCREEN(.IE,JA.);
    T:=T+SCREEN(.I,JE.) +2*SCREEN(.I,J.) +SCREEN(.I,JA.);
    T:=T+SCREEN(.IA,JE.)+  SCREEN(.IA,J.)+SCREEN(.IA,JA.);
    T:=ROUND(T/10);
/* yeni pixel değerinin eşik değerlerini aşıp aşmadığının
   kontrol edilmesi */
    IF T>9 THEN T:=9;
    IF T<0 THEN T:=0;
/* hesaplanan değerin buffer'a konması. Zira bu
   değerin hemen yerine yazılması yanlış olacaktır. */
    BUFFER(.I,J.):=T;
  END;
END.
```

FİLTRELER

ORİJİNAL GÖRÜNTÜ

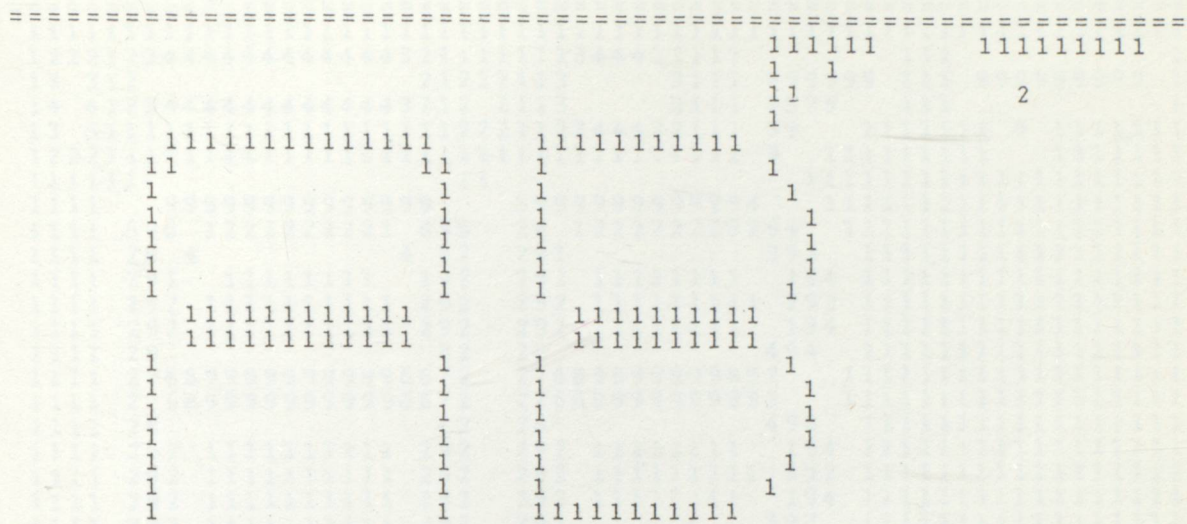


NOT: Her rakkam bir pixelin gri renk tonunu , boşluk olarak görülen bölgeler ise beyaz pixelleri temsil etmektedir. AB Karakterlerinin üstünde görülen bölgede algılama hatalarından doğan gürültü vardır. Bunlar filtrenin hem karakterlere hem de gürültüye cevabını gösterebilmek amacıyla özellikle konulmuştur.



KERNEL MATRİSİ: -1 -1 -1
                 -1   8 -1
                 -1 -1 -1 şeklindedir.(LAPLACIEN)

Bu görüntü normal LAPLACIAN filtresi kullanıldıktan sonra elde edilmiştir. Gürültülerin de abartıldığına dikkat edin. Ayrıca, zemin renginin de değişmediği gözlenmektedir.

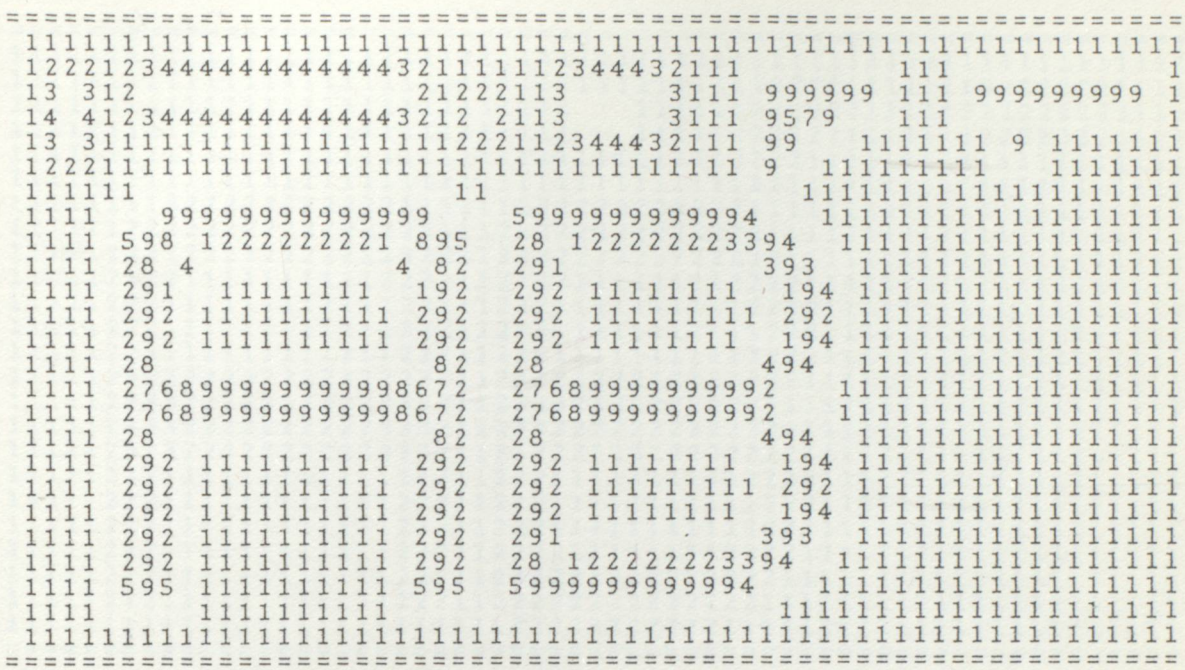


KERNEL MATRİSİ: -0.1 -0.1 -0.1

-0.1 0.8 -0.1

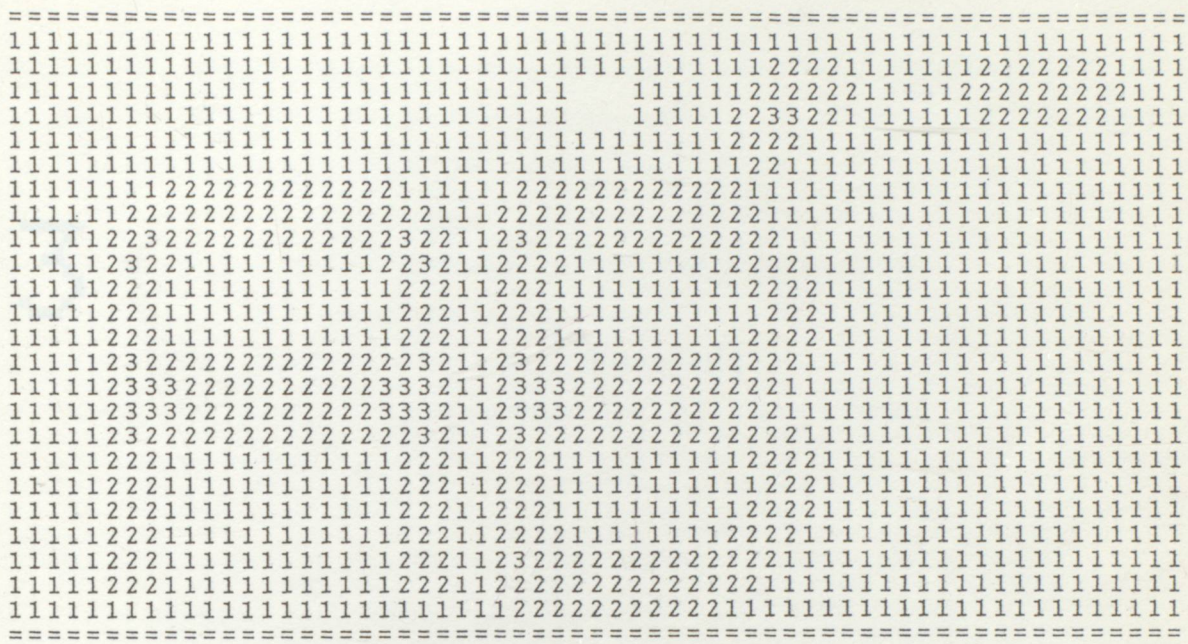
-0.1 -0.1 -0.1 şeklindedir.

Bu, normal LAPLACIAN filtresi kullanıldıktan sonran x'=x/10 formülüyle point process de uygulanılmasına karşılık düşer. Gürüntü silikleşmiştir. Fakat buna karşın küçük bazı gürültülerin tamamen yok olduğu gözlenmektedir.



KERNEL MATRİSİ: -1 -1 -1
                 -1   9 -1
                 -1 -1 -1 şeklindedir.

Bu, normal LAPLACIAN filtresinin biraz değiştirilmiş bir halidir. Frekansın sıfır olduğu bölgeler filtrelenmeyip orijinal değerlerinde bırakılarak zemin renginin değişmemesi sağlanmıştır. Böylece yalnız kenarların keskinleştirilmesi sağlanmış ve orijinal renklere sadık kalınmıştır.



KERNEL MATRİSİ: 0.1 0.1 0.1

0.1 0.2 0.1

0.1 0.1 0.1 şeklindedir.

Bu, Kernel alçak frekans filtresine karşılık gelir ve gürültü yok etmede kullanılır. Fakat neyin gürültü neyin görüntü olduğu kesin olmadığından görüntü bulanıklaşır ve geniş gürültü bölgeleri filtrelenemeyebilir.

3

ŞABLON EŞLEME YÖNTEMİYLE KARAKTER TANIMA

Şablon eşleme yöntemiyle karakter tanıma basit bir şekilde tanımlanmak istenilirse şu soru sorulmalıdır. 'Görüntü daha önceden tanımlanan bir nesne içeriyor mu?' Bu soru türü TEMPLATE MATCHING olarak adlandırılır. Bu yöntemde, önceden tanımlanmış bir dijital şekil (şablon) sistematik olarak bütün sayfa boyunca dolaştırılır. Şablondaki her pixelin değeri sayfadaki, kendisine karşı düşen pixelın değerine eşit olduğu zaman görüntüdeki cismi tesbit etmiş oluruz. Tarama sırasında sayfa biterse o zaman bu cisim görüntüde yoktur denilir. Genelde bir görüntüde bir cisim aranıldığı için şablon sayfadan küçük olur. Yani matematiksel olarak şablonun domeni, orijinal resmin domeninden daha küçüktür diyebiliriz.

Daha gerçekçi olmamız gerekirse şablonun, görüntüye ne kadar uyduğundan bahsetmemiz gerekir. Zira hiç bir görüntü okuyucu bu sunuç için mükemmel değildir.

g(i,j) dijital şeklin pixel değeri
t(i,j) şablonun pixel değeri
D şablonun domeni (Boyutları) ise.

Şablonun (m,n) noktasında orijinal görüntüye ne kadar uyduğu;

$$M(m,n) = \sum_{\substack{i,j \text{ such} \\ \text{that} \\ (i-m,j-n) \\ \text{is in } D.}} \sum |g(i,j) - t(i-m, j-n)|.$$

formülü ile hesaplanır. Burda domen dışına çıkmamak gerekmektedir. Her şablon tek tek kullanılarak en az hata oluşturan aranmalıdır. Bunun yanında iki kriterden daha bahsedebiliriz. Bunlar,

$$E(m,n) = \left\{ \sum_i \sum_j [g(i,j) - t(i-m, j-n)]^2 \right\}^{1/2}$$

ve

$$S(m,n) = \max_{i,j} |g(i,j) - t(i-m, j-n)|$$

Bunlardan iki vektör arasındaki EUCLIDEAN farkı olarak bilinen ikinci formülü daha yakından incelersek. Karekökleri ve kare alma işlemlerini basitleştirmek

amacıyla aşağıdaki fomülün taraf tarafa karesini alsak ve kare alma işlemlerini açarsak,

$$E^2(m,n) = \sum_i \sum_j [g^2(i,j) - 2g(i,j)t(i-m, j-n) + t^2(i-m, j-n)],$$

elde ederiz. Buradan E nin en küçük olma hali incelenirse:

Son bileşenin toplamı şablonun tüm şekil boyunca dolaştırıldığı göz önüne alındığında sabit bir sayı olarak karşımıza çıkacaktır. Birinci bileşen görüntünün enerjisi olarak bilinir, ve bunun şimdilik ihmal edilebilecek kadar küçük olduğunu kabul edersek hatanın minimum olma hali ortadaki bileşenin en büyük olması durumunda olacaktır. Yani

$$R_{gt}(m,n) = \sum_i \sum_j g(i,j)t(i-m, j-n),$$

olarak g ve t fonksyonları arasında tanımlanan CROSS-CORRELATION'un büyük olması durumunda bunlar arasında benzerlikten bahsedebiliriz. Fakat tamamı beyaz olan bir görüntü üzerinde şablon yüksek benzerlik gösterebilir. Bu nedenle normalize edilmiş Cross-Correlation kullanmak gerekecektir. Bu yüzden,

$$N_{gt}(m,n) = \frac{1}{\left[\sum_i \sum_j g(i,j)^2\right]^{1/2}} R_{gt}(m,n),$$

formülü daha iyi sonuç verecektir.

Öte yandan probleme BAYES karar verme teorisi ile yaklaşırsak.

Bayes kuralı:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)},$$

$$p(x) = \sum_{j=1}^{s} p(x \mid \omega_j)P(\omega_j).$$

g(i,j) dijital şekil
r(i,j) 1. şablon
s(i,j) 2. şablon
p dijital şeklin orijinale benzeme olasılığı ise ve

$$P(g \mid \omega_r) = \prod_{i,j} p^{1-|g(i,j)-r(i,j)|}(1-p)^{|g(i,j)-r(i,j)|}$$

$$P(g \mid \omega_s) = \prod_{i,j} p^{1-|g(i,j)-s(i,j)|}(1-p)^{|g(i,j)-s(i,j)|}.$$

her iki formülün taraf tarafa logaritmasını alırsak.

$$\log P(g \mid \omega_r) = \left[\log\left(\frac{1-p}{p}\right)\right] \sum_{i,j} |g(i,j) - r(i,j)| + \sum_{i,j} \log p$$

$$\log P(g \mid \omega_s) = \left[\log\left(\frac{1-p}{p}\right)\right] \sum_{i,j} |g(i,j) - s(i,j)| + \sum_{i,j} \log p.$$

elde ederiz. p en azından 1/2 den büyük olduğundan log((1-p)/p) da daima negatif olacaktır. Ayrıca log p lerin toplamı da sabit olduğundan.

$$\sum_{i,j} |g(i,j) - r(i,j)| < \sum_{i,j} |g(i,j) - s(i,j)|$$

elde ederiz.

Minimum hata elde etmek için bu iki sonuçtan en küçük olanını seçeriz. Eğer birinci hata olasılığı küçük ise şekil g, şablon r'ye daha fazla benzeyecektir. Aynı şekilde eğer ikinci hata olasılığı daha küçük ise şeklin, s şablonuna benzeme ihtimali daha yüksek olacaktır. Fakat şimdiye kadar incelediğimiz her yötemde |g(i,j)-t(i,j)| gibi bir fark terimi işin içine girmektedir ki bu çeşitli problemler doğurabilir. Örneğin şablon renk tonu ile dijital görüntünün tonu uyuşmayabilir ve bu durumda fark terimi yüzünden pixellerin değerleri birbirlerinden çıkartılarak toplandığında elde edilecek hata sıfır olmayacaktır. Böylece ton farkı göz önüne alınmadığında tamamen aynı olan şekillerde uyuşmazlık varmış gibi sonuçlar alınacaktır. Bu problemi gidermek için her defasında POINT PROCESS gerekecektir. Bu da algoritmaya

azımsanamayacak bir yük getirecektir. Kanımca bu yöntemler yerine iki vektör arasındaki CORRELATION COEFFICIENT faktörü kullanılırsa ve elde edilen sonucun mutlak değeri kullanılırsa Point Process yapılmayabilir. Böylece algoritmanın renk tonlarından meydana gelecek hatadan etkilenmemesi sağlanır. Öteyandan eğer şablonun boyu ile görüntüde aranılan cismin boyu aynı değilse o zaman yapılması gereken işler arasına Geometric Process de girecektir. Fakat bildiğimiz gibi ölçek değiştirme bazı detayların kaybolmasına da sebep olabilir. Bu yüzden algoritmalarda yanılma payından bahsedilmelidir.

CORRELATION YÖNTEMİYLE ŞEKİL TANIMA

N adet görüntü gözü (CELL) içeren N boyutlu bir u vektörü göz önüne alalım. Burada her cell bir veya daha fazla pixel'e karşılık düşmektedir. u Vektörünün her bir elemanı kendisine karşı gelen cell'in parlaklık değerini içermektedir. Doğal olarak bunlar ayrık tip, bir birleşik olasılık yoğunluk fonksiyonuna sahip rasgele değişkenler olacaktır.

Eğer $u(X_1, X_2, \ldots, X_n)$ birleşik olasılık yoğunluk fonksiyonları $f(x_1, x_2, \ldots, x_n)$ ve uzayı R olan ayrık tip rastgele değişkenlerin bir fonksiyonu ise,

$$E[u(X_1, X_2, \ldots, X_n)] = \sum_{(x_1, \ldots, x_n)} \cdots \sum u(x_1, x_2, \ldots, x_n) f(x_1, x_2, \ldots, x_n),$$

eğer oluşursa buna

$$u(X_1, X_2, \ldots, X_n).$$

fonksiyonunun beklendik değeri denir. Ve;

Eğer $u_1(X_1, X_2, \ldots, X_n) = X_i$ ise

$E[u_1(X_1, X_2, \ldots, X_n)] = E(X_i) = \mu_i$

X_i 'nin ortalama değeri olarak adlandırılır.

Eğer $u_2(X_1, X_2, \ldots, X_n) = (X_i - \mu_i)^2$, ise

$E[u_2(X_1, X_2, \ldots, X_n)] = E[(X_i - \mu_i)^2] = \sigma_i^2 = \mathrm{Var}(X_i)$

X_i 'nin variance değeri olarak adlandırılır.

Eğer $u_3(X_1, X_2, \ldots, X_n) = (X_i - \mu_i)(X_j - \mu_j), i \neq j,$

$E[u_3(X_1, X_2, \ldots, X_n)] = E[(X_i - \mu_i)(X_j - \mu_j)] = \sigma_{ij} = \mathrm{Cov}(X_i, X_j)$

X_i ve X_j 'in covariance değeri olarak adlandırılır.

Eğer o ve o standart sapmalar pozitif ise

$$\rho_{ij} = \frac{\mathrm{Cov}(X_i, X_j)}{\sigma_i \sigma_j} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

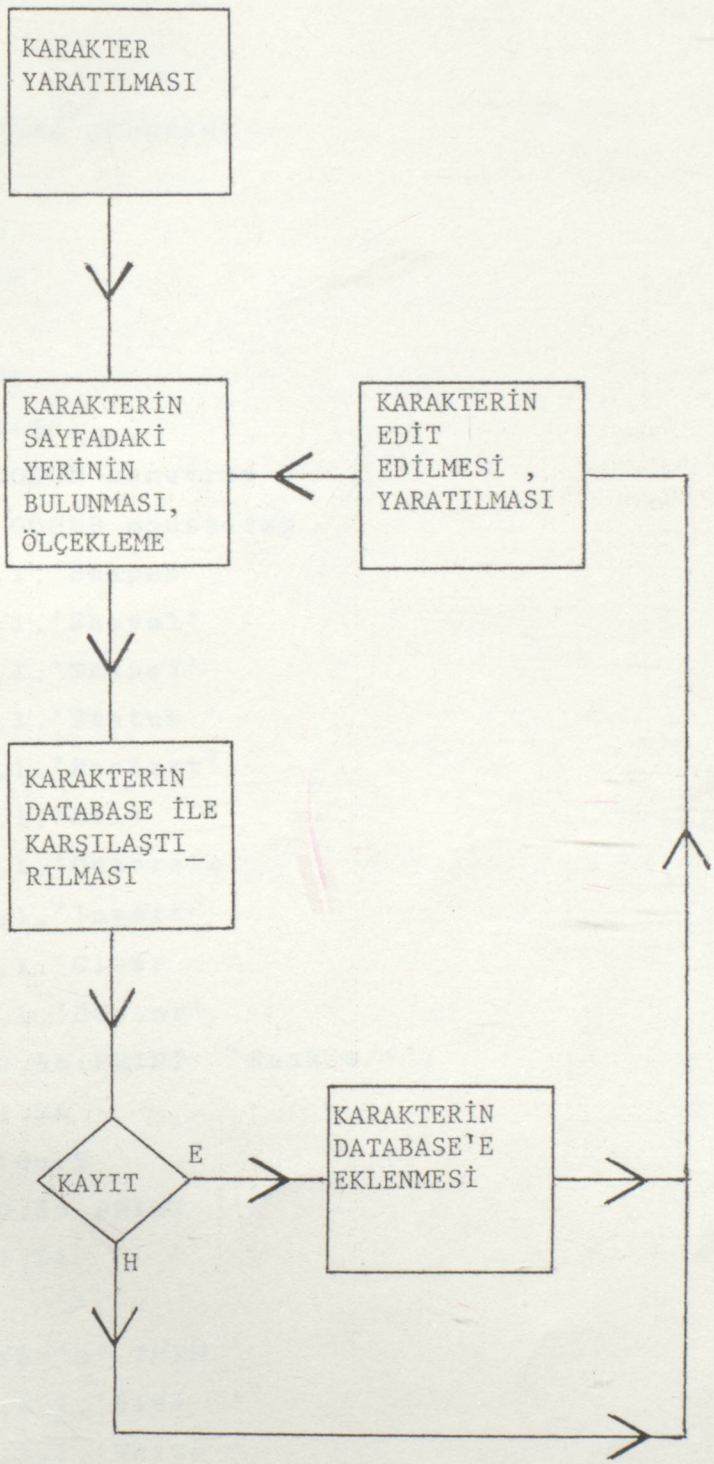
X_i ve X_j 'in arasındaki korolasyon katsayısı olarak adlandırılır.

X_i 'nin ortalama değer ve varyansı ya birleşik olasılık yoğunluk fonksiyonundan yada marjinal olasılık yoğunluk fonksiyonundan hesaplanabilir. Böylece CORRELATION COEFFICIENT faktörü olarak

$$R = \frac{[1/(n-1)] \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{[1/(n-1)] \sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{[1/(n-1)] \sum_{i=1}^{n} (Y_i - \bar{Y})^2}}$$

elde edilir.

Öte yandan FOURIER transformları da şekil tanıma konusunda geniş yer tutar. Fakat bu uygulamada hız önemli olduğundan Fourier transformlarının performansı etkilemesi göze alınamamiştır.



ŞEKİL TANIMA AKIŞ DİYAGRAMI

ŞEKİL TANIMA PROGRAMI

```
DEFINT a-z
DEFDBL q
DIM a(600)
DIM corro(49)
ON MENU GOSUB menutrap
ON MOUSE GOSUB mousetrap
MENU 1,0,1,'Shapes'
MENU 1,1,1,'Shape1'
MENU 1,2,1,'Shape2'
MENU 2,0,1,'Status '
MENU 2,1,1,'Restart'
MENU 2,2,1,'Halt   '
MENU 3,0,1,'Generate'
MENU 3,1,1,'Insert  '
MENU 3,2,1,'Clear   '
MENU 4,0,1,'DColor'
LOCATE 10,56:PRINT  'Renk e/*';
LOCATE 11,56
INPUT colour$
LOCATE 10,56:PRINT  '          ';
LOCATE 11,56
PRINT  '    ';
IF colour$='e' THEN
  MENU 4,1,1,'Blue  '
  MENU 4,2,1,'White '
  MENU 4,3,1,'Black '
  MENU 4,4,1,'Orange'
```

```
                ELSE
   MENU 4,1,1,'White '
   MENU 4,2,1,'L-Gray'
   MENU 4,3,1,'D-Gray'
   MENU 4,4,1,'Black '
   PALETTE 0,1,1,1
   PALETTE 1,.66,.66,.66
   PALETTE 2,.33,.33,.33
   PALETTE 3,0,0,0
                END IF
MENU ON
CLS
WHILE 1
SLEEP
WEND
menutrap:
menuid=MENU(0)
menuitem=MENU(1)
ON menuid GOTO shapes,status,generation,dotcolor
RETURN
shapes:
shape$='Shape'+CHR$(menuitem+48)
shape=menuitem
  RETURN
status:
ON menuitem GOTO restart,halt
RETURN
restart:
  RUN
halt:
  COLOR 1,0
  END
generation:
```

```
ON menuitem GOTO insert,clrscrn
RETURN
insert:
generation=1
MOUSE ON
RETURN
clrscrn:
IF generation<>1 THEN RETURN
LINE (0,0)-(xstart,ystart),dotcolor,bf
LINE (0,0)-(xstart,ystart),1,b
LINE (617-xdot,0)-(617,ydot),dotcolor,bf
LINE (617-xdot,0)-(617,ydot),1,b
zemin=dotcolor
RETURN
dotcolor:
dotcolor=menuitem-1
COLOR dotcolor,dotcolor
RETURN
mousetrap:
IF generation<>1 THEN RETURN
IF MOUSE(0)=0 THEN RETURN
MOUSE OFF
MENU 3,1,0,'Insert  '
CLS
check0:
xstart=MOUSE(1):ystart=MOUSE(2)
IF xstart>550 THEN xstart=550
LINE (0,0)-(xstart,ystart),1,b
check1:
IF INKEY$='1' THEN check2
IF MOUSE(0)=0 THEN check1
LINE (0,0)-(xstart,ystart),0,b
GOTO check0:
```

```
check2:
ydot=FIX(ystart/9)+1
xdot=FIX(xstart/10)+1
LINE (617-xdot,0)-(617,ydot),1,b
IF INKEY$='2' THEN check3
IF MOUSE(0)=0 THEN check2
xpoint=MOUSE(1)
ypoint=MOUSE(2)
IF xpoint>xstart-7 THEN check2
IF ypoint>ystart-7 THEN check2
ychar=FIX(ypoint/9.1)+1
xchar=FIX(xpoint/10.1)+1
LOCATE ychar,xchar:PRINT ' ';
PSET (617-xdot+xchar,ychar),dotcolor
GOTO check2
check3:
generation=0
MENU 3,1,2,'  Insert'
MENU ON
COLOR 3,0
LOCATE 6,57
xsinir=618-xdot
ysinir=ydot-1
REM GET (xsinir,1)-(616,ysinir),a
LOCATE 9,56:PRINT '??????';
FOR ustsatir=1 TO ysinir
   FOR j=xsinir TO 616
   IF POINT(j,ustsatir)<>zemin THEN cik1
   NEXT j
NEXT ustsatir
cik1:
FOR altsatir=ysinir TO 1 STEP -1
    FOR j=xsinir TO 616
```

```
    IF POINT(j,altsatir)<>zemin THEN cik2
    NEXT j
NEXT altsatir
cik2:
FOR sagsutun=616 TO xsinir STEP -1
    FOR i=1 TO ysinir
    IF POINT(sagsutun,i)<>zemin THEN cik3
    NEXT i
NEXT sagsutun
cik3:
FOR solsutun=xsinir TO 616
    FOR i=1 TO ysinir
    IF POINT(solsutun,i)<>zemin THEN cik4
    NEXT i
NEXT solsutun
cik4:
LINE (solsutun-1,ustsatir-1)-(sagsutun+1,altsatir+1),,b
genislik=sagsutun-solsutun+1
yukseklik=altsatir-ustsatir+1
qgenadim=genislik/7
qyukadim=yukseklik/7
qsutunbas=qgenadim-INT(qgenadim)
qsatirbas=qyukadim-INT(qyukadim)
qmakspoint=qgenadim*qyukadim
alfpoint=1
ttoplam=0
t2toplam=0
xyaz=580
FOR qyavassutun=solsutun TO sagsutun STEP qgenadim
yyaz=40
   FOR qyavassatir=ustsatir TO altsatir STEP qyukadim
   toplam=0
      FOR qhizlisutun=qsutunbas TO qgenadim-1
```

```
            FOR qhizlisatir=qsatirbas TO qyukadim-1
            xx=qyavassatir+qhizlisatir
            yy=qyavassutun+qhizlisutun
            toplam=toplam+POINT(yy,xx)
            NEXT qhizlisatir
         NEXT qhizlisutun
         yenipoint=INT(toplam/qmakspoint+.1)
         IF yenipoint>3 THEN
            yenipoint=3
            toplam=3*qmakspoint
            PRINT toplam
            END IF
         ttoplam=ttoplam+toplam
         t2toplam=t2toplam+toplam^2
         corro(alfpoint)=toplam
         alfpoint=alfpoint+1
         qsutunbas=INT(qyavassutun)-qyavassutun
         qsatirbas=INT(qyavassatir)-qyavassatir
         yyaz=yyaz+1
         PSET (xyaz,yyaz),yenipoint
      NEXT qyavassatir
      xyaz=xyaz+1
NEXT qyavassutun
PRINT
qoldqr=0
oldchar$=' '
OPEN 'corro' FOR INPUT AS Ö9
   WHILE NOT EOF(9)
      xycarpim=0
      FOR i=1 TO 49
         INPUT Ö9,sayi
         xycarpim=xycarpim+sayi*corro(i)
      NEXT i
```

```
        INPUT Ö9,ytoplam
        INPUT Ö9,ykare
        INPUT Ö9,ycharŞ
        xtoplam=ttoplam
        xkare=t2toplam
        qust=ABS(xycarpim-xtoplam*ytoplam/49)
    qalt=(xkare-xtoplam*xtoplam/49)*(ykare-ytoplam*ytoplam/49)
        qr=qust/SQR(qalt)
        LOCATE 9,56
        deneme=INT(qr*100+.4)
        PRINT deneme;ycharŞ;' '
        IF qr>qoldqr THEN
             qoldqr=qr
             oldcharŞ=ycharŞ
             LOCATE 8,56
             PRINT deneme;ycharŞ;' '
             END IF
      WEND
      LOCATE 9,56:PRINT 'Sonuc '
CLOSE Ö9
LOCATE 11,56:PRINT 'Kayit ';:LOCATE 12,56:INPUT kayitŞ
LOCATE 11,56:PRINT '       ';:LOCATE 12,56:PRINT '       ';
IF kayitŞ='e' THEN
   LOCATE 1,1
   LOCATE 11,56:PRINT 'Bu ne ';:LOCATE 12,56:INPUT buneŞ
   LOCATE 11,56:PRINT '       ';:LOCATE 12,56:PRINT '       ';
   OPEN 'corro' FOR APPEND AS Ö9
   FOR i=1 TO 49
      WRITE Ö9,corro(i)
   NEXT i
   WRITE Ö9,ttoplam
   WRITE Ö9,t2toplam
   WRITE Ö9,buneŞ
```

```
  CLOSE Ö9
END IF
GOTO check2
END
```

ŞABLON DATABASE'İNİN KONTROL EDİLMESİ

```
DEFINT a,i,t,k,x,y
DIM a(49)
PALETTE 0,1,1,1
PALETTE 1,.66,.66,.66
PALETTE 2,.33,.33,.33
PALETTE 3,0,0,0
CLS
xstart=12
ystart=8
xchar=2
INPUT 'file';fileŞ:CLS
OPEN fileŞ FOR INPUT AS Ö9
WHILE NOT EOF(9)
maks=0
   FOR i=1 TO 49
      INPUT Ö9,a(i)
      IF a(i)>maks THEN maks=a(i)
   NEXT i
   maks=3/maks
   INPUT Ö9,toplam
   INPUT Ö9,kare
   INPUT Ö9,charŞ
   LOCATE 3,xchar:PRINT  charŞ;
   FOR i=0 TO 6
   ii=i*7
      FOR j=1 TO 7
      PSET (xstart+i,ystart+j),INT(a(ii+j)*maks)
```

```
      NEXT j
   NEXT i
   xstart=xstart+10
   xchar=xchar+1
WEND
CLOSE Ö9
```

SONUÇ

Projemde GÖRÜNTÜ OKUMA VE ŞEKİL TANIMA çalışmalarına değişik bir bakış yapılmış olup, sistemin kendisinden istenilen başarıyı, zaman içinde, belirli bir alanda uzmanlaşarak ve hatalarından ders alarak sağlaması yoluna gidilmiştir. Bu sistemle ilk uygulama olarak yazı karakter ve sembollerini, okumaya, işlemeye ve tanımaya yönelik bir çalışma yapılmış olup tatmin edici sonuçlar elde edilmiştir. Bu sonuçlar, sistem kendi konusunda uzmanlaştıkça iyileşmekte ve doğruya daha yakın tahminler olarak ortaya çıkmaktadır.

Yazı karakterlerini tanıyan sistem her ne kadar kullanıcının el yazısına bağımlıysa da görüntü işleme yardımcı programları ve kataloğun birden fazla el yazısı içerebilme özelliği kullanılarak, bir çok kişinin el yazılarını ayırdedebilmek mümkün olmaktadır. Fakat görüntü işleme yardımcı programları (doğal olarak) bazı ayırdedici ayrıntıların yok olmasına sebep olabilmektedir. Bu nedenle her zaman bir yanılma payı bulunmaktadır ve bu ancak performanstan fedakarlık yapılarak giderilebilir. Öte yandan bir sayfada ortalama 2000 karakter bulunduğunu ve en az 20 şablonun gerektiği göz önüne alınırsa aslında performansın ne derece önemli olduğu görülür. Bu yüzden hız ile kesinlik arasında kabul edilir bir denge sağlanması şarttır.

Sonuç olarak çalışmam, domene daha az bağımlıdır ve her alandaki uygulamalar için yaklaşım yapabilmektedir. Ayrıca devamlı öğrenme içinde bulunması ve kendini her an geliştirebilmesi projemin diğer bir özelliğidir.

SÖZLÜK

-DIGITIZE: Analog video sinyalleri genliklerinin dijital değerlere çevirilmesi işlemi.

-DISPLAY: Televizyonun yaptığı gibi elektriksel sinyalleri optik sinyallere (görüntüye) dönüştüren cıhaz.

-FILTER: Görüntü işlemede, şeklin parlaklığını veya boyutunu değiştiren işlem.

-FRAME: Şekli oluşturan görüntü satırlarının tamamı.

-FRAME BUFFER: CPU'nun ve DISPLAY ünitesinin aynı anda erişebildiği, görüntü saklamak için dizayn edilmiş yüksek hızda bellek alanı.

-FRAME MEMORY: Bellek haritasında görüntü saklamak için kullanılan herhangi bir bölge.

-FRAME STORAGE: Bir görüntüyü geçici olarak saklamak için kullanılan çok hızlı bellek alanı.

-IMAGE PROCESSING: Tanıma veya belirginleştirme amacıyla bir şeklin nokta nokta analiz edilmesi.

-INTENSITY: Görüntünün bir noktasındaki ışık parlaklığı.

-INTERLACE: Standart video formatında tek ve çift numaralı görüntü satırlarının dönüşümlü olarak görüntülenmesi.

-LOOKUP TABLE: Pixel değerlerinin değiştirilmesi için kullanılan Hardware veya Software tablo.

-PIXEL: Dijital görüntünün en küçük elemanı. Enine ve boyuna bir şekilde adreslenebilen bellek alanı yapısındadır.

-RESOLUTION: Görüntü işlemede, bir pixel için ayrılan ve gri renk tonu yaratılmasında kullanılan bit adedi.

KAYNAKLAR

- Pattern Classification and Scene Analysis
  Wiley-Interscience Publication 1973
  Richard O. Duda , Peter E. Hart

- Image Pattern Recognition
  Springer-Verlag New York Inc. 1980
  V.A. Kovalevsky

- İstatistik Teorisine Giriş
  Yıldız Üniversitesi 1989
  Kirkor Harutunyan

- Probability and Statistical Inference
  Macmillan Publishing Co.,Inc. 1983
  Robert V. Hogs, Elliot A. Tanis

- Pattern Recognition a Statistical Approach
  Prentice-Hall International,Inc. 1982
  P.A. Devijver, J. Kittler

- Desktop Publishing
  BİT Büro - İnformatik Telekomünikasyon-Mart 1987
  Mustafa Doğrusoy

- Introduction to Image Processing Algorithms
  Byte The Small Systems Journal-March 1987
  Benjamin M. Dawson

- Low-Cost Image Processing
  Byte The Small Systems Journal-March 1987
  Charles McManis

- Finding the TITANIC
  Byte The Small Systems Journal-March 1986
  Ben Dawson

- Binary-Image-Manipulation Algorithms in the Image View Facility
  I.B.M. Journal Res. Develop.-January 1987
  K.L. Anderson, F.C. Mintzer, G. Goertzel, J.L. Mitchel
  K.S. Pennington, W.B. Pennebaker

- PANDA: Processing Algorithm for Noncoded Document Acquisition
  I.B.M. Journal Res. Develop.-January 1987
  Yi-Hsin Chen, Frederick C. Mintzer, Keith S. Pennington

- Artificial Intelligence
  Computer Vision
  John Mayhew, John Frisby

- C Made Easy
  McGraw Hill
  H.Schildt

- ASM86 Language Reference manual
  Intel Corporation 1981

- I.B.M. PC AT Technical Reference
  I.B.M. Corporation 1984

- Inroduction to the Commodore AMIGA 2000
  Commodore-Amiga,Inc.

- Image Scanner/Interface IX-12/IX21F Service Manual
  Canon Inc.

- Image Scanner Image Library
  Canon Inc.

- Amiga Basic
  Commodore-Amiga,Inc.

- Pascal/VS Reference
  I.B.M. Corp.

ÖZGEÇMİŞ

Meir ESKİNAZİ, 1965 yılında İstanbul'da doğdu.Liseyi Özel Beyoğlu M. Lisesi'nde bitirdi.1986'da Yıldız Üniversitesi Mühendislik Fakültesi Bilgisayar Bilimleri Mühendisliği Bölümünden mezun oldu. Aralık-1986'da Netaş Bilgi İşlem Departmanında Sistem Programcısı olarak çalışmaya başladı ve Ocak-1989'a kadar bu görevini sürdürdü.