TÜRKİYE CUMHURİYETİ

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ASSEMBLY ILE GORUNTU ISLEME

Öğrenci Numarası:

21011011

Ad-Soyad: Turabi Yıldırım

E-Posta Adresi:turabi.yildirim@std.yildiz.edu.tr

Telefon:05394330186

Alt Seviye Programlama Dersi Raporu

Turabi Yıldırım

8 Oca,2023

İÇERİK

MODÜL TANITIMI

1- Görüntü İşleme Sonuçları

1.1 Dilation

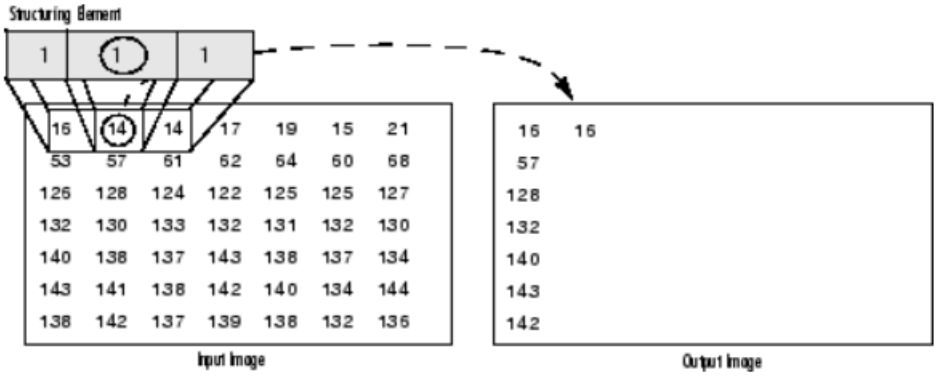
Görüntünün ilk hali 5x5 filtre ile dilation 10x10 filtre ile dilation

1.2 Erosion

Görüntünün ilk hali 5x5 filtre ile erosion 10x10 filtre ile erosion

2- KOD ACIKLAMALARI

2.1 Kodun mantığıyla ilgili ana temalar



Filtreleme işlemim buradaki 1*3 lük arrayle yapılan filtreleme işleminin aynısı.

Fakat böyle filtreleme yapınca sadece yatay olarak görüntü değiştiriliyor bu yüzden aynı resmi filtreledikten sonra 90 derece çevirip tekrar filtreledim ve bu sayede 3*3 bir matris ile filtreliyormuş gibi bir sonuç aldım.

En sonda ise resmi 3 kere daha 90 derece çevirdim ve bu şekilde bir matrisle filtrelemiş gibi sonuç elde ettim.

Filtrele Filtrele

Tekrar döndürmek için 3 kere çevirme

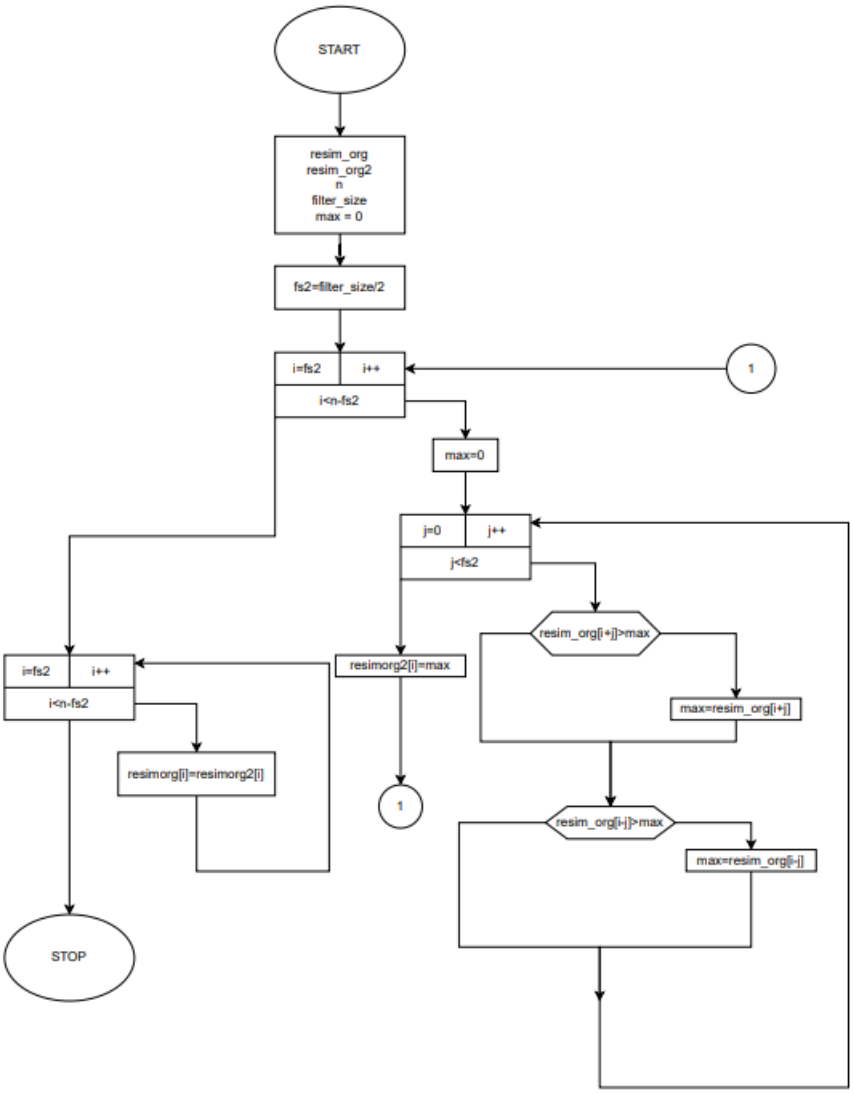
İkinci sorun ise filtreye sokulmuş resmin bilgilerini saklayacak ikinci bir matris olmamasıydı ve bunu ise short içindeki ilk 8 biti kullanarak çözdüm.

Short içindeki ilk 8 bit yeni koyulan noktalar son 8 bit ise eski noktalar olarak kullandım ve en sonda 8 bit sağa kaydırıp yeni resim noktalarına ulaştım.

Resmi çevirirken resimdeki tüm i değerlerini i/width + (i%width) * width değeri ile değiştirdim.

Çeviriken de aynı şekilde ilk 8 bit yeni koyulan noktalar son 8 bit ise eski noktalar olarak kullandım.

2.2 Kodun Flowchartı



Burada max ı min(256) yapıp resimorg[i+j]<min şeklinde arama yaparsak da erosion versiyonunu elde ederiz.

Çevirme işlemini ise flowchart kullanmadan ekledim.

Bu flowchart parçasına filter() dersek program şu şekilde çalışıyor.

Filter()

Çevirme()

Fiter()

Çevirme()Çevirme()Çevirme()

2.3 Assembly Kodunun Açıklaması

Filtreleme Kodu

```
MOV EAX, filter_size
SHR EAX, 1; EAX = i
MOV ECX, n
SUB ECX, filter_size
INC ECX
OUTERLOOP :
PUSH ECX
    XOR EBX, EBX; max
    XOR EDX, EDX; j
    MOV ECX, filter_size
    SHR ECX, 1
    INC ECX
    INNERLOOP :
```

Burada flowcharttaki gibi for loopların variablelarını ayarlıyorum ve max variableını 0 lıyorum.

```
ADD ESI, EAX
ADD ESI, EAX
ADD ESI, EDX
ADD ESI, EDX
PUSH ECX
MOV CX,[ESI]
SHL CX,8
SHR CX,8
CMP CX, BX
POP ECX
```

Burada resim_org[i+j] variableına erişip SHL8 sonra SHR 8 yapip ilk 8 bitini elde ediyorum sonra ilk 8 bitle elimdeki max degerini karşılaştırıyorum.

```
JLE NEXT
XOR EBX, EBX
PUSH ECX
MOV CX, [ESI]
SHL CX, 8
SHR CX, 8
MOV BX,CX
POP ECX
NEXT:
```

Burada eğer maxdan büyükse resim_org[i+j]] variableının tekrar ilk 8 bitini aynı yöntemle elde ediyorum sonra ilk 8 biti BX in (max) içine yazıyorum.

```
MOV ESI, resim_org
    ADD ESI, EAX
    ADD ESI, EAX
    SUB ESI, EDX
    SUB ESI, EDX
    PUSH ECX
    MOV CX, [ESI]
    SHL CX, 8
    SHR CX, 8
    CMP CX, BX
    POP ECX
    JLE NEXT2
    XOR EBX, EBX
    PUSH ECX
    MOV CX, [ESI]
    SHL CX, 8
    SHR CX, 8
    MOV BX, CX
    POP ECX
    NEXT2:
```

Aynı işlemi resim_org[i-j] variable ı için tekrar ediyorum.

```
INC EDX
    LOOP INNERLOOP
```

Burada j değerini arttırıyorum LOOP kısımlarını gerçekleştiriyorum.

```
    SHL BX,8
    MOV ESI,resim_org
    ADD ESI,EAX
    ADD ESI,EAX
    PUSH EAX
    MOV AX,[ESI]
    ADD AX,BX
    MOV [ESI],AX
    POP EAX
    INC EAX
    POP ECX
DEC ECX
JNS OUTERLOOP
```

Burada BX değerimi 8 sola kaydırarak AX ile topluyorum ve ilk 8 bite max değerimi(resim_org2[i]) son 8 bitte ise eski değerimi resim_org[i] tutuyorum.

EAX değeri i olduğu için onu arttırıyorum.Burada LOOP OUTERLOOP kullanmadım çünkü LOOPUN range i yetmedi.

Bazı kısımlarda elimde register kalmadığı için PUSH POP kullandım (burada PUSH EAX POP EAX)

```
MOV ECX,n
SUB ECX, filter_size
INC ECX
MOV EAX, filter_size
SHR EAX, 1; EAX = i
ASSIGNLOOP:
MOV ESI, resim_org
ADD ESI, EAX
ADD ESI, EAX
MOV BX, [ESI]
SHR BX, 8
MOV [ESI], BX
INC EAX
LOOP ASSIGNLOOP
```

En sonda ise elimdeki değerlerin hepsini 8 bit sağa kaydırıyorum ve elimde sadece ilk 8 bit kalıyor.

90 Derece Çevirme Kodu

```
; yan cevirme yeri
MOV ECX, n
MOV EAX, 0
YANCEVIRMELOOP_4:
MOV ESI, resim_org
ADD ESI, EAX
ADD ESI, EAX
XOR EBX, EBX
MOV BX, [ESI]
SHL BX, 8
SHR BX, 8
MOV ESI, resim_org
```

İlk başta loopa giriyorum ve resim[i] değerinin ilk 8 bitini alıyorum.

EAX = i , ECX = n, BX= resim[i] değeri

```
    PUSH ECX
    PUSH EAX
    ; ECX bulma kismi
    XOR ECX, ECX
    ECXBULMALOOPU_4 :
INC ECX
    MOV EAX, n
    XOR EDX, EDX
    DIV ECX
    CMP ECX, EAX
    JNE ECXBULMALOOPU_4
    XOR EDX, EDX
    POP EAX
    PUSH EAX
```

Burda yerine koyacağım resim[i/width + (i%width) * width] değerine ulaşmak için kullanacağım ECX EAX değerlerini pushluyorum.

ECX bulma loopunda ise EAX i bölümüyle eşit olana kadar tüm sayılara

bölüyorum.(Karekök bulmak için)Burada EAX i kullandığım için POPlayıp pushluyorum ve eski i değerine döndürüyorum.

```
DIV ECX
ADD ESI, EAX
ADD ESI, EAX
MOV EAX, EDX
MUL ECX
ADD ESI, EAX
ADD ESI, EAX
```

Burada ESI değerine (i/width + (i%width) * width)*2 değerini ekliyorum.

```
POP EAX
POP ECX
XOR EDX, EDX
MOV DX, [ESI]
SHL DX, 8
ADD BX, DX
MOV ESI, resim_org
ADD ESI, EAX
ADD ESI, EAX
MOV[ESI], BX
INC EAX
LOOP YANCEVIRMELOOP 4
```

EAX ve ECX değerlerini eski hale döndürüyorum sonra yeni değerin bitlerini alıp sağa 8 bit kaydırıyorum.

Eski değer resim i BX de saklıydı.

Bu değerleri topluyorum artık yeni değer ilk 8 bitte eski değer son 8 bitte.

Sonra bu değeri resim[i] yerine yazıyorum.

```
    MOV ECX, n
    MOV EAX, 0
    ASSIGNLOOPYAN_4 :
MOV ESI, resim_org
    ADD ESI, EAX
    ADD ESI, EAX
    MOV BX, [ESI]
    SHR BX, 8
    MOV[ESI], BX
    INC EAX
    LOOP ASSIGNLOOPYAN_4
```

Burda ise tüm değerlerimi 8 bit sağa kaydırıyorum ve sadece ilk 8 bit olan kısım elimde kalıyor.

Filter()

Çevirme()

Filter()

Çevirme()

Çevirme()

Çevirme()

En son kodları bu şekilde tekrarlayıp filtreleme işlemini tamamlıyorum.

KAYNAKÇA

1: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html

TURABİ YILDIRIM-21011011