TÜRKİYE CUMHURİYETİ

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BÖLÜMÜ

ALT SEVİYE PROGRAMLAMA ÖDEV 2

Öğrenci No:20011017

Ad-Soyad: Ömer Diner

E-Posta Adresi:omer.diner@std.yildiz.edu.tr

Telefon:05

Alt Seviye Ödev 2 Raporu

Dr. Erkan
USLU

14 Ocak,2022

GİRİŞ

Morphology is known as the broad set of image processing operations that process images based on shapes. It is also known as a tool used for extracting image components that are useful in the representation and description of region shape.

The basic morphological operations are:

- Erosion
- Dilation

Dilation:

- Dilation expands the image pixels i.e. it is used for expanding an element A by using structuring element B.
- Dilation adds pixels to object boundaries.
- **The value of the output pixel is the maximum value of all the pixels in the neighborhood.** A pixel is set to 1 if any of the neighboring pixels have the value 1.

Erosion:

- Erosion shrink-ens the image pixels i.e. it is used for shrinking of element A by using element B.
- Erosion removes pixels on object boundaries.:
- **The value of the output pixel is the minimum value of all the pixels in the neighborhood.** A pixel is set to 0 if any of the neighboring pixels have the value 0.

Kaynak: https://www.geeksforgeeks.org/

Ödevimi buradaki kaynaktan yararlanarak yaptım. Kırmızı renklendirilmiş yazıda söylenen uygun piksel değerini işlem tipine göre seçme işlemini erosion ve dilasion için gerçekleştirdim. Yeni hesaplanan piksel değerlerini orijinal diziye koydum. Hesaplamalar yapmak içinse orijinal değerleri bozmadan kullanmak gerektiği için ek bir dizi kullandım.

KODLAR VE AÇIKLAMALAR

Dilation

Pseudocode

```
procedure Dilation(n: int, filter_size: int, resim_org: short[])
    resim = new short[n]
    for i = 0 to n-1
        resim[i] = resim_org[i]
    end for
    for y = filter_size/2 to n-1 - filter_size/2
        for x = filter_size/2 to n-1 - filter_size/2
            max_val = 0
            for i = -filter_size/2 to filter_size/2
                for j = -filter_size/2 to filter_size/2
                    if resim[(y+i)*n + (x+j)] > max_val
                        max_val = resim[(y+i)*n + (x+j)]
                    end if
                end for
            end for
            resim_org[y*n+x] = max_val
        end for
    end for
    print("Dilation islemi sonucunda resim \"dilated.pgm\" ismiyle
olusturuldu...")
end procedure
```

-Baştaki for dışındaki n'ler kök n olarak alınmalı.

```
64
65  void Dilation(int n, int filter_size, short* resim_org) {
66
67      short* resim = (short*)malloc(n * sizeof(short));
68      int i;
69      for (i = 0; i < n; i++)
70          resim[i] = resim_org[i];
71
72      __asm {
73
74          MOV EDI,resim_org
75          MOV ESI,resim
76          MOV EBX,filter_size
77          SHR EBX,1
78          XOR EDX,EDX
79          MOV EAX,1026
80          MUL EBX
81          MOV EBX,EAX
82          MOV ECX,513
83          SUB ECX,filter_size
84
85  LOOP_1 :
86          PUSH ECX
87          PUSH EBX
88          MOV ECX,513
89          SUB ECX,filter_size
```

67-70 arası orijinal değerleri kaybetmemek için ek dizi tanımlıyorum.
Data segmentte belli bir konuma yani dizinin indexlerine ulaşmak için ESI, EDI veya EBX kullanmak gerekli olduğu için orijinal resmin adresini EDI'ya, yedek resmin adresini ise ESI'ya atıyorum.

BX'te ise filtre boyutu var. İşlemi yaparken pikseli ortalayacağımız için kenar değerlerini kullanamayacağız. Bunun için döngülerde orayı atlamamız lazım. Filtre boyutunu SHR EBX,1 komutu ile yarıya bölüyorum. EDX değerini sıfırlamak için XOR işlemi kullandım. Çünkü birkaç satır sonrasında çarpma işlemi yapılmış, beklenmeyen bir sonuç gelmemesi için EDX'in sıfırlanması lazım. Resmin boyutu 512*512 olduğu için manuel olarak kullandım bu bilgiyi. Tek bir kenarı bulmak için n değerinin karekökünü almadım. 82-83. satırlar en dıştaki döngünün sınırlarını belirliyor.83'te fitre boyutu çıkarılıyor çünkü sağdan ve soldan filtre/2 kadar eksik piksel gezilecek.

En içteki döngü hariç her döngüye girdiğimde ECX ve EBX değerlerini stacke atıyorum ki değerler kaybolmasın.

Resim normalde iki boyutlu bir yapı ama parametre olarak tek boyutlu bir dizi geldiği için filtre boyutuna göre uygun satırlara ulaşmak için ayarlamalar yapmak gerekiyor , bu yüzden 4 adet döngü kullandım toplamda. İlk ikisi her bir pikseli geziyor gibi düşünebiliriz. En içteki iki ise o piksele filtrenin denk düşürdüğü konumları alıyor.

```
90
91  LOOP_2 :
92          PUSH ECX
93          PUSH EBX
94          MOV EAX,1026
95          XOR EDX,EDX
96          MOV ECX,filter_size
97          SHR ECX,1
98          MUL ECX
99          SUB EBX,EAX
100         MOV ECX,filter_size
101         MOV AX,0
102
103 LOOP_3 :
104         PUSH ECX
105         PUSH EBX
106         MOV ECX, filter_size
107
108 LOOP_4 :
109         CMP WORD PTR[ESI + EBX],AX
110         JNA SMALLER
111         MOV AX, WORD PTR[ESI + EBX]
112
```

Resmin piksellerinin göreceli konumlarına [ESI+EBX] ile ulaşıyorum. ESI resmin temel adresi EBX ise benim ayarlamalar yaparak filtrenin denk geleceği yerlere ulaşmamı sağlıyor.

LOOP2 sonunda AX'e 0 değerini atıyorum çünkü her piksel için çevresindeki maksimum değere sahip pikseli bulmamız lazım. Bunun için karşılaştırma işlemi yapacağız ve en büyük değeri bulacağız. O yüzden karşılaştırma yapacağımız değerin olabilecek en küçük değer olması gerekiyor. Resimde de 0'dan küçük piksel değeri olamayacağı için AX'e 0'ı verdim.
LOOP 4 içinde karşılaştırma işlemi yapılıyor. O anki piksel değeri AX'ten büyükse AX'e onun değeri geçiyor.

```
113 SMALLER:
114         ADD EBX,2
115         LOOP LOOP_4
116
117         POP EBX
118         ADD EBX,1024
119         POP ECX
120         LOOP LOOP_3
121
122         POP EBX
123         POP ECX
124         MOV WORD PTR[EDI + EBX],AX
125         ADD EBX,2
126         LOOP LOOP_2
127
128
129         POP EBX
130         ADD EBX,1024
131         POP ECX
132         LOOP LOOP_1
133
134     }
135     printf("\nDilation islemi sonucunda resim \"dilated.pgm\" ismiyle olusturuldu...\n");
136 }
```

Bu son etikette , her döngü için parametrelere gerekli değerler ekleniyor ve LOOP 4,3,2,1 komutu ile sırası ile döngülere giriliyor. İç içe bir yapı olduğu için en önce LOOP4 geliyor. Döngülere girmeden önce değerleri push ile stack'e atmıştık. Onları da ters sıra ile geri çekiyorum.

124. satırda filtre uygulanmış işlem sonucunu orijinal dizinin uygun indisine yazıyorum.

LOOP2 VE LOOP4'ün son satırlarında EAX değil de AX kullanma sebebim işlem yapılırken dizinin elemanlarının boyutu ile kullanılan register boyutunun eşit olmasını sağlamak.

Bu dört döngü sonrasında her kenardan filtre boyu/2 eksik eleman kadar içeriden olmak üzere tüm elemanlar için çevresindeki en büyük eleman bulunuyor ve depolanıyor.

AAAAAA
ABBBBA
ABBBBA
ABBBBA
AAAAAA

Mesela üst taraftaki harfleri bir resmin temsili gibi kabul edersek ve 3*3 filtre uygulamak istersek hiçbir A harfi filtrenin merkezine tam oturamayacağından değerleri değişmeyecektir. B'ler ise güncellenerekten yazılacaktır.

Dilation fonksiyonu sonrası oluşan resimler

Filtre boyutu 3

Filtre boyutu 7

Erosion

Bu işlemde gezeceğimiz piksellerin etrafındaki minimum değeri alacağız. Onun dışında bir değişiklik yok.

```
138 void Erosion(int n, int filter_size, short* resim_org) {
139
140     short* resim = (short*)malloc(n * sizeof(short));
141     int i;
142     for (i = 0; i < n; i++)
143         resim[i] = resim_org[i];
144
145     __asm {
146         MOV EDI,resim_org
147         MOV ESI,resim
148         MOV EBX,filter_size
149         SHR EBX,1
150         XOR EDX,EDX
151         MOV EAX,1026
152         MUL EBX
153         MOV EBX,EAX
154         MOV ECX,513
155         SUB ECX,filter_size
157   LOOP_1 :
158             PUSH ECX
159             PUSH EBX
160             MOV ECX,513
161             SUB ECX,filter_size
162
163   LOOP_2 :
164             PUSH ECX
165             PUSH EBX
166             MOV EAX,1026
167             XOR EDX,EDX
168             MOV ECX,filter_size
169             SHR ECX,1
170             MUL ECX
171             SUB EBX,EAX
172             MOV ECX,filter_size
173             MOV AX,7FFFh
```

173.satırda AX'e registere atılabilecek en yüksek pozitif değeri verdim. Çünkü minimum değeri ararken en başta kontrol amaçlı olan değişkenin dizideki değerlerden büyük olması gerekir.

```
175  LOOP_3:
176          PUSH ECX
177          PUSH EBX
178          MOV ECX,filter_size
179
180  LOOP_4 :
181          CMP WORD PTR[ESI + EBX],AX
182          JA BIGGER
183          MOV AX,WORD PTR[ESI + EBX]
184
185  BIGGER :
186          ADD EBX,2
187          LOOP LOOP_4
188
189          POP EBX
190          ADD EBX,1024
191          POP ECX
192          LOOP LOOP_3
193
194          POP EBX
195          POP ECX
196          MOV WORD PTR[EDI + EBX], AX
197          ADD EBX, 2
198          LOOP LOOP_2
199
200          POP EBX
201          ADD EBX, 1024
202          POP ECX
203          LOOP LOOP_1
204
205      }
206      printf("\nErosion islemi sonucunda resim \"eroded.pgm\" ismiyle olusturuldu...\n");
207  }
```

181.satırda yapılan kontrol işlemi sonucu eldekinden daha küçük bir değer bulunursa 183.satırda AX güncelleniyor.

196.satırda orijinal resmin piksel değerine AX'in son hali yazılıyor , yani filtre uygulanmış ve sonuç olarak etrafındaki minimum elemanın değeri.

Erosion Fonksiyonu Sonrası Oluşan Resimler

3*3 filtre

7*7 filtre