



**YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ**

Bilgisayar Mühendisliği Bölümü

EROSION – DILATION
ASSEMBLY IMPLEMENTASYONU
ALT SEVİYE PROGRAMLAMA ÖDEV II

Öğretim Görevlisi:

FURKAN ÇAKMAK

Ödevi hazırlayan:

HARUN OKTAY

Öğrenci Numarası: 20011080

İletişim: Harun.oktay@std.yildiz.edu.tr

İstanbul 2022

İÇİNDEKİLER

GİRİŞ.....	2
EROSION-DILATION.....	2
Anahtar Kelimeler.....	2
Algoritma Çalışma Mantığı.....	3
→Filtrenin Görsel Üzerinde İlerleme Dinamiği.....	3
→Filtre İçinde Pixelleri Gezinme Dinamiği.....	3
→Bulunan Değerleri Nereye Yazmalı?.....	3
KOD TANITIMI.....	4-5
ÇIKTILAR.....	6-8
SONUÇ.....	8
KAYNAKÇA.....	9

GİRİŞ

Görüntü işleme yöntemlerinden olan “Dilation” ve “Erosion” uygulamalarını, alt seviye dillerinden biri olan “80x86” programlama dilinde programladım. Bu komutları, hazır verilen “C++” kodu içerisine Inline olarak entegre ettim. Bu raporumda temel olarak kod parçacığının nasıl çalıştığını ve programın nasıl çıktıları ortaya çıkardığını göstereceğim.

EROSION - DILATION

“Erosion” ve “Dilation” en çok kullanılan morfolojik görüntü işleme operasyonlarından ikisidir. Bu operasyonlar temel olarak bir filtre matrisinin görüntü pikselleri üzerinde gezinerek, maximum ya da minimum noktaları bulup görüntü üzerine işlenmesi işlemini yaparlar (bu uygulamamda grayscale görüntüler üzerinde çalıştığım için piksellerin aralığı 0-255 arasında olacaktır.). Erosion filtre içerisindeki minimum (siyaha yakınsayan), Dilation ise filtre matrisi sınırları içerisindeki maximum(beyaza yakınsayan) değeri bularak, görüntüye işler. Erosion ve dilation operasyonlarının kullanım alanlarından bazılarını sıralayayım:

- ➔ Görüntü üzerindeki gürültüleri azaltma amaçlı.
- ➔ Nesneleri çevreden ayırıştırarak tespit etmek.
- ➔ Yoğunluk oluşan noktaları ya da delikleri tespit etmek.

Göreceğiniz üzere bu iki operasyon temel olarak görüntüleri analiz etmek amaçlı kullanılıyor. Bazı zamanlarda görüntü sentezlemek için de kullanılır.

ANAHTAR KELİMELER

80x86 Assembly, Görüntü işleme, Dilation, Erosion

ALGORİTMA ÇALIŞMA MANTIĞI

Filtrenin Görsel Üzerinde İlerleme Dinamiği

512x512 bir görüntü üzerinde gezinen bir filtre düşündüm. Bu filtre boyutunu kullanıcıdan alıyorum. Bu filtre görüntünün sol üst köşesinden başlayarak, sola doğru ilerliyor, sütun durma noktasına geldiği zaman, bir alt satıra inerek sütunları sıfırlıyor ve tekrardan sütun durma noktasına gelene kadar ilerliyor. Döngü değişkenim sıfırlanana kadar bir önceki cümlede anlattığım prosesi yapmaya devam ediyor. Filtrenin pixel matrisi üzerinde gezinme dinamiği bu biçimde

Filtre İçindeki Pixelleri Gezinme Dinamiği

Filtre indisi, Filtre matrisinin kapsadığı alan içinde boyutunun karesi kez gezinerek ihtiyacı olan değeri (maximum ya da minimum) buluyor. İlk olarak satırları üzerinde sırasıyla ilerliyor. filter_size değişkenimin 5 olduğunu varsayarsak 5 kez gezdikten sonra sütun sıfırlanıyor ve bir sonraki satıra geçiş yapıyor. Kaçınıcı satırda olduğumu tutan ayrı bir kontrolüm mevcut. Eğer satır sayısı 5 olursa bu döngüden çıkacak ve üst paragrafta anlattığım gibi filtrenin konumu değişmiş olacak ya da algoritmadan çıkış yapmış olacağız.

Bulunan Değeri Nereye Yazmalıyız?

Genel olarak bulunan max-min değerler, aynı boyutta yeni bir görsel matrisi açılarak filtrenin merkezinin eşdeğer konumuna yazdırılır. Ancak bu algoritmam ek dizi kullanmadığı için, bulunan değerleri filtrenin sol üst köşesine yazdırıyor. Bir sonraki adımda filtre matrisinin kapsadığı alan içerisinde sol üst köşenin bulunmadığını, algoritma çalışma mantığına bakarak anlayabiliriz. Sonuç olarak sol üst köşe bulunan değerleri yazdırmak için en uygun yer.

KOD TANITIMI

İki kod da birbiriyle neredeyse aynı çalıştığı için Erosion ve Dilation kardeş operasyonlar olarak bilinir. Erosion ve Dilation kodları arasındaki tek fark eden nokta filtre matrisinin içindeki maximum ya da minimumu bulan dallanma komutu farklılığı. Dilationda JA ($Op1 > Op2$) iken, Erosion'da JB ($Op1 < Op2$) olarak kullanıldı. Bu dallanma komutu farkı filtrenin (maximum ya da minimum) hangi değeri bulacağını belirliyor.

DILATION- EROSION

```
void Dilation-Erosion(int n, int filter_size, short* resim_org) {
```

Fonksiyonun aldığı parametreler. **filter_size** → Kullanıcıdan alınan filtre boyutu
resim_org → Resmin ilk pixelinin adresini tutan pointer

```
__asm {  
    MOV EDI, resim_org ; Resmi EDI registerına taşıdım. Kod boyunca  
                        ; resim pixellerine EDI ve ESI üzerinden  
                        ; erişim sağlayacağım.  
  
    MOV ECX, 512  
    SUB ECX, filter_size  
    INC ECX  
    MOV EAX, ECX  
    PUSH EAX  
    MUL ECX  
    MOV ECX, EAX  
    POP EAX  
    SHL EAX, 1  
    PUSH EAX  
  
    MOV ESI, EDI  
    MOV EAX, [EDI]  
    MOV EBX, 0  
    MOV EDX, 0  
}
```

En dışta bir LOOP(for) oluşturdum
CX içerisinde resimdeki
pixel sayısı hesaplanarak
oluşturduğum çıkış değeri var.
[(512-filter_size+1)x(512-filter_size+1)]
EAX ← (512-filter_size+1)
Bitler sola 1 kez kaydırılırsa
değer 2 ile çarpılır.

WALK: EDI içerisinde resim olduğunu
hatırlayalım. aynıysa artık ESI içinde
Filtrenin ilk değerini EAX'e koydum
i, j ← 0 gibi düşünelim.

```

CTRL:      CMP EAX, [ESI]           ;Max noktayı bulan temel karşılaştırma
           (JB ya da)JA STEADY      ;Filtre matrisindeki diğer değerlerle karşılaştırıyorum.
           MOV EAX, [ESI]          ;Daha büyük bir değer ( daha açık ton)
                                           ;varsa yerine koy.

STEADY:     ADD ESI, 2              ;Yoksa satır gezinmeye devam et.
           INC EBX                 ;indisi artır.
           CMP EBX, filter_size    ;Filtrenin sonuna geldik
           JE SUTUN               ;Artık ya sütunu artıracamız. ya da sona geldik.
           JMP CTRL

SUTUN:      MOV ESI, EDI           ;Filtrenin ilk değerine döndük
           INC EDX                 ;EDX ile sütun indisi durma koşulu kontrolü
           CMP EDX, filter_size    ;yapıyoruz eğer sütun da 3 olmuşsa 0 1 2
           JE FILTER              ;gezmişiz demektir, filtre konumu
           PUSH EAX                ;değişir FILTER etiketine atlar
           PUSH EDX
           MOV EAX, 1024           ;Eğer filtre sütun sonuna gelmediyse:
           MUL EDX
           POP EDX
           ADD ESI, EAX            ;filtrenin ilk değerine (1024 x indisi) kadar
           POP EAX                ;ekleyerek satırını gezeceğimiz
           MOV EBX, 0              ;sütunu set ediyoruz.
           JMP CTRL               ;Satırını gezmeye başla.

FILTER:     POP EBX                ;Filtrenin konumunu değiştiren etikete geldik.
           MOV [EDI], EAX          ;resme bulduğumuz değeri yerleştiriyoruz.
           ADD EDI, 2              ;Filtreyi iki byte/ bir word arttır.
           CMP EDI, EBX            ;Filtrenin gezeceği satır sonu mu?
           JNE CONTINUE           ;Değilse filtrenin konumu hazır. CONTINUE.
           MOV EBX, EDI            ;Satır Sonuna geldik,
           ADD EBX, 1024           ;Filtrenin durma noktasını 1 sütun aşağı almak için
           PUSH EBX                ;Filtre satır durma noktası.
           MOV EAX, filter_size    ;EAX 2*(filtre boyutu-1)
           DEC EAX                 ;Burada matris gibi düşünmemek gerekiyor
           SHL EAX, 1              ;Filtreme EAX kadar değer ekliyorum.
           ADD EDI, EAX            ;Fazladan push yapmak istemeyiz.
           JMP ENDROW

CONTINUE:   PUSH EBX               ; PUSH Filtre durma noktası.
                                           ; FILTER etiketinde POPlanmak üzere.

ENDROW:     LOOP WALK              ;CX sıfırlanana kadar devam et.
           POP EAX                 ;Ne kadar PUSH, o kadar POP.
}

```

```

printf("\nDilation islemi sonucunda resim \"dilated.pgm\" ismiyle
olusturuldu...\n");
}

```

EROSION:original→3→5→7 lena.pgm DILATION:original→3→5→7



0



3



5



7





original lena

1



bir kez 7x7 filtre ile erosion uygulanmış lena.pgm

2



iki kez 7x7 filtre ile erosion uygulanmış lena.pgm

3

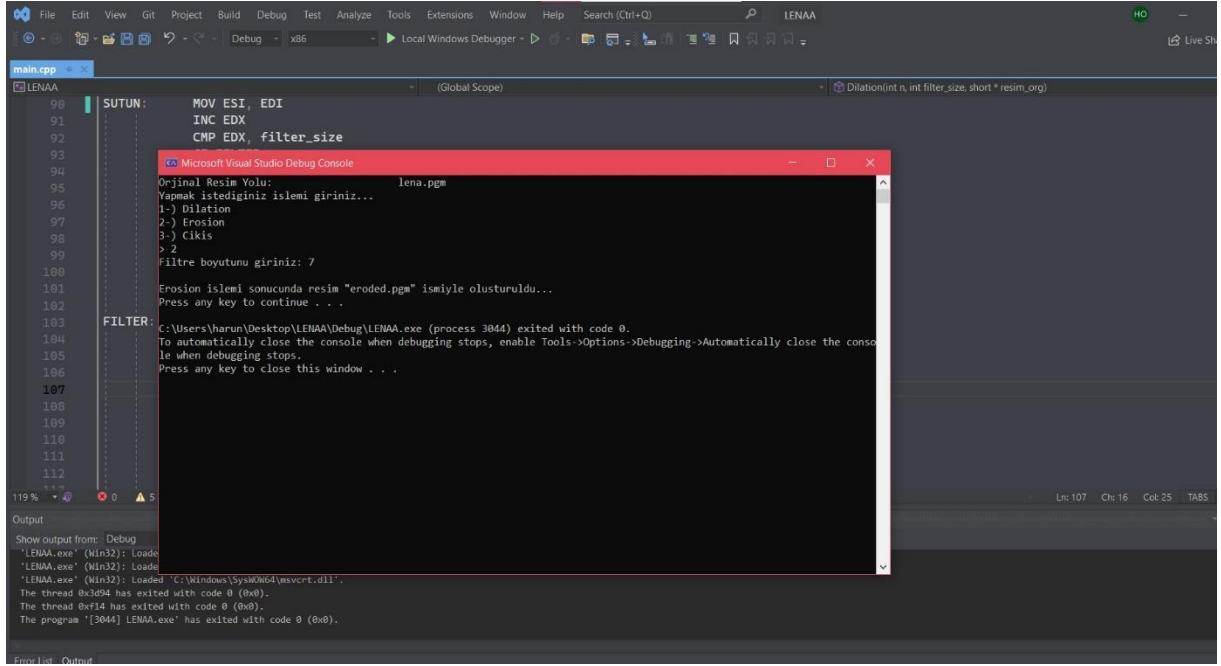


üç kez 7x7 filtre ile erosion uygulanmış lena.pgm

Çıktılardan göreceğimiz üzere Erosion işlemi bize daha koyu, Dilation işlemi bize daha açık renkli bir çıktı oluşturuyor. Bu işlemler astronomi, biyoloji, görüntü tanıma gibi uygulamalarda çok işlevsel olabiliyor. Çıktılardan göreceğiniz üzere bu operasyonlar belirli noktaları(max min) daha da ön plana çıkarıyor. Herhangi bir nesneyi, çevresinden farklı bir noktayı tespit etmek için kullanılabilir.

Filtre mantığını anlatırken max- min noktaları sol üst köşeye yazdığımdan bahsetmiştim. Bu nedenle resimlerin sağ kısımlarında ve alt kısımlarında işlenmemiş noktalar mevcut. Filtre büyüklüğü arttıkça işlenmemiş kısımlar da büyüyor.

MENÜ



```
90 SUTUN: MOV ESI, EDI
91 INC EDX
92 CMP EDX, filter_size
93
94
95 Original Resim Yolu: lena.pgm
96 Yapmak istediğiniz işlemi giriniz...
97 1-) Dilatation
98 2-) Erosion
99 3-) Cıkis
100 > 2
101 Filtre boyutunu giriniz: 7
102 Erosion işlemi sonucunda resim "eroded.pgm" ismiyle olusturuldu...
103 Press any key to continue . . .
104 FILTER: C:\Users\harun\Desktop\LENAA\Debug\LENAA.exe (process 3044) exited with code 0.
105 To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
106 Press any key to close this window . . .
107
108
109
110
111
112
113
114
115
116
117
118
119 % 0 5
Output
Show output from: Debug
'LENAA.exe' (Win32): Loaded 'C:\Windows\System32\user32.dll'.
'LENAA.exe' (Win32): Loaded 'C:\Windows\System32\user32.dll'.
The thread 0x3054 has exited with code 0 (0x0).
The thread 0x3054 has exited with code 0 (0x0).
The program '[3044] LENAA.exe' has exited with code 0 (0x0).
```

SONUÇ :

Alt seviye programlama dili olan assembly ve yüksek seviyeli dilleri beraber kullanmak, oldukça verimli sonuçlar ortaya çıkarabilir. İşlemleri assembly üzerinden yürütmek bize zaman ve işlem gücü tasarrufu sağlar. Bu uygulamada olduğu gibi yüksek sayıda veriyi işlememiz gerekiyorsa ya da analog zamana olabildiğince yakın bir sonuç elde etmek istiyorsak (FPS oyunlarda ateş etme örnek verilebilir.), programın bu kısmını assembly kullanarak gerçekleştirmek, programın kalitesini oldukça artıracaktır.

KAYNAKÇA:

- [1] [https://en.wikipedia.org/wiki/Erosion_\(morphology\)](https://en.wikipedia.org/wiki/Erosion_(morphology))
→ https://en.wikipedia.org/wiki/File:Grayscale_Morphological_Erosion.gif
- [2] https://www.researchgate.net/figure/An-Example-of-Gray-scale-Dilation-and-Erosion_fig3_33760747
- [3] <https://stackoverflow.com/questions/1472768/implementing-erosion-dilation-in-c-c>
- [4] <https://www.hlevkin.com/hlevkin/06testimages.htm>
- [5] https://www.youtube.com/watch?v=sqJJ3FKs_0M
- [6] *80x86 Instruction Set*