YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

ALT SEVİYE PROGRAMLAMA DERSİ
ÖDEV 2 RAPORU

Öğrenci No: 20011007
Öğrenci Ad-Soyad: Bilal Müftüoğlu
E-Posta: bilal.muftuoglu1@std.yildiz.edu.tr

İÇERİK

Şekil 1 Dilated 3x3

Şekil 2 Eroded 3x3

Şekil 3 Dilated 5x5

Şekil 4 Eroded 5x5

Şekil 5 Dilated 7x7

Şekil 6 Eroded 7x7

Şekil 7 Dilated 20x20

Şekil 8 Eroded 20x20

2- Dilation C++ kodu

Dilation işleminin C++ kodu karşılığı aşağıdaki gibidir. Önce bu kodu daha sonra ise Assembly kodu karşılığını yazmamın sebebi raporda Assembly kodunu daha rahat anlatabilmekdir.

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

    int half_f_size = filter_size / 2;
    short* resim_copy = (short*)malloc(n * sizeof(short));

    int row_size = sqrt(n);

    // resim_org'un her pikseli için
    for (int i = 0; i < n; i++) {

        int max_pixel = 0;
        // filtre çevresindeki piksellerin kontrolü
        for (int j = -half_f_size; j <= half_f_size; j++) {
            for (int k = -half_f_size; k <= half_f_size; k++) {
                // resmin sınırlarını aşmaması için
                if ((i + j * row_size + k >= 0) && (i + j * row_size + k < n)) {
                    // en büyük pikselin bulunması
                    if (resim_org[i + j * row_size + k] > max_pixel) {
                        max_pixel = resim_org[i + j * row_size + k];
                    }
                }
            }
        }
        resim_copy[i] = max_pixel;
    }

    // resim_copy --> resim_org kopyalama
    for (int i = 0; i < n; i++) {
        resim_org[i] = resim_copy[i];
    }

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

3- Dilation assembly kodu ve anlatımı

```
void Dilation(int n, int filter_size, short* resim_org) {
    __asm {
            //resmin kare oldugu bilindigi icin satir veya sutun sayisi n'nin
            //karekoku hesaplanarak bulunur. islem sonunda ecx = satir sayisi
            mov ebx, n
            mov ecx, 0
            mov edx, 0
        iter:
            inc ecx
            mov eax, ecx
            mul ecx
            cmp eax, ebx
            jge done
            jmp iter
        done : ;//ecx = sqrt(n) = satir sayisi
```

Dilation fonksiyonunun başı yukarıdaki gibidir. Burada yapılan işlem karekök hesaplama işlemidir. Parametre olarak gelen n sayısının karekökü hesaplanır. Bu sayede satır veya sütun boyutu bulunur. İşlem sonunda sonuç ecx'de bulunur.

```
            mov esi, 0; // esi = i
        start2:
            cmp esi, n; // i == n
            je exit2; // dongu sona erer
            mov edx, filter_size
            shr edx, 1;// filter_size/2
            mov ebx, edx
            neg ebx;// ebx = j = -filter_size/2
            mov eax, 0; // eax = max
```

C++ kodunda görüldüğü üzere iç içe 3 tane for döngüsü vardır. Yukarıdaki kodda ise en dıştaki for döngüsü başlatılır. Bu döngüde n uzunluklu resim_org dizisinin her elemanının yeni değeri hesaplanır. Öncelikle i (esi) değerine 0 atanır. Daha sonra döngü her başa sardığında i değeri n'den küçük mü diye kontrol edilir. Eğer küçük değilse program sona erer küçükse bir içteki for döngüsü için j (ebx) değişkenine –(filtre boyutu/2) değeri atanır. Örneğin filtre boyutu 3 ise j değeri -1 olur ve 1 olana kadar devam eder. Max (eax) değişkeni de döngü başlamadan 0'lanır.

```
        start3:
            mov edx, filter_size;
            shr edx, 1;// filter_size/2
            cmp ebx, edx; // j == filter_size/2
            jg exit3; // dongu sona erer
            mov edi, edx
            neg edi; // edi = k = -filter_size/2
```

Yukarıda görüldüğü gibi 2. for döngüsüne girdikten sonra j değeri (filtre boyutu/2) ‘ den büyükse döngü sona erer küçük eşitse 3. for döngüsü için k (edi) değişkenine –(filtre boyutu/2) değeri atanır. J değişkeni filtre matrisinde gezindiğimiz değerlerin merkezdeki değerden satır uzaklığını belirtirken K değişkeni sütun uzaklığını belirtmektedir.

```
        start4 :
            mov edx, filter_size
            shr edx, 1;// filter_size/2
            cmp edi, edx; // k == filter_size/2
            jg exit4; // dongu sona erer
```

Yukarıdaki kısımda en içteki for döngüsünün kontrolü yapılır. K (edi) değişkeni (filtre boyutu/2) ‘den büyükse döngü sona erer değilse devam edilir.

```
            push ebx;// j degerini korumak icin push islemi
            push eax;// max degerini korumak icin push islemi
            mov eax, ecx;// eax = satir sayisi
            xor edx, edx
            mul ebx;// eax = j * satir sayisi
            add eax, esi;// eax = i + j * satir sayisi
            add eax, edi; // eax = i + j * satir sayisi + k

            cmp eax, 0;// Eger eax 0'dan kucukse dizi disina cikildigindan devam
edilmez
            jl exit5
            cmp eax, n; // Eger eax n'ye buyuk esitse dizi disina cikildigindan
devam edilmez
            jge exit5
```

C++ kodunda da görüldüğü üzere dizi dışına çıkmamak için bir kontrol yapılmaktadır. Hesapladığımız piksel yani filtre matrisinin

merkezindeki pikselin dizideki yeri i idi. J ise filtre matrisinde o anda baktığımız pikselin merkez piksel ile satır farkıydı. K'da sütun farkıydı. (i + j*satır sayısı + k) hesaplaması ile matriste karşılaştırmak istediğimiz pikselin dizideki yerini bulabiliriz. Yukarıdaki kodda öncelikle değerini korumak istediğimiz bazı yazmaçlara push işlemi yapıyoruz. Daha sonra (i + j*satır sayısı + k) değerini eax'e alıyoruz. Dizi dışına çıkılmaması için bu değerin 0'dan büyük eşit n'den küçük olması gerekiyor. Bu durumların kontrolleri yapılmaktadır ve sağlamıyorsa çıkış yapılır.

```
            pop ebx;// stack'e en son atilan max degeri ebx'e cekilir
            //ama stack'de de durmasi istenildiginden tekrar push yapilir
            push ebx
            xchg eax, ebx;// eax = max , ebx = i + j * satir sayisi + k
            push esi;// i degerini korumak icin push islemi
            mov esi, resim_org
            shl ebx, 1
            cmp word ptr[esi + ebx], ax;// resim_org[i + j * satir sayisi + k] ==
max
            jle exit6;//eger dizide kontrol edilen deger max'dan kucuk esitse devam
edilmez
            mov ax, word ptr[esi + ebx];// yeni max degeri ax'e alinir
```

Yukarıdaki kodda son olarak en içteki if kontrolü yapılır. Eğer filtre matrisinde baktığımız piksel değeri merkezdekinden büyük ise max (ax) değişkenine yeni değer atanır.

```
        exit6:
            shr ebx, 1
            pop esi
            pop edx;// cop deger
            pop ebx
            inc edi;// k++
            jmp start4
```

En içteki if kontrolünden sonra üstteki koda gelinir. Gerekli pop işlemleri yapıldıktan sonra k (edi) değişkeni 1 arttırılır ve en içteki for döngüsünün başına dönülür.

```
        exit5 :
            pop eax
            pop ebx
            inc edi; // k++
            jmp start4; // basa don
```

En içteki if'den bir önceki if kontrolünde şart sağlanmazsa üstteki kısıma gelinir. Gerekli pop işlemleri yapıldıktan sonra k (edi) değişkeni 1 attırılır ve en içteki for döngüsünün başına dönülür.

```
        exit4:
            inc ebx; // j++
            jmp start3; // basa don
```

En içteki for döngüsü bittikten sonra j (ebx) değişkeni 1 attırılır ve 2. for döngüsünün başına dönülür.

```
        exit3:
            push ax // max elemani bulma islemi bitti, max deger stack'e atilir
            inc esi; // i++
            jmp start2; // basa don
```

2. for döngüsü bittikten sonra filtre matrisindeki maksimum değer bulunmuş olur. Bu değer stack'e atılır ve i (esi) değişkeni 1 attırıldıktan sonra ilk for döngüsünün başına dönülür.

```
        exit2:
            mov ecx,n // tum donguler sona erdi
            mov edi,n // stackten max degerler teker teker alinip diziye sondan
basa dogru eklenir
            mov esi,resim_org
        son:
            dec edi
            pop ax
            shl edi,1
            mov word ptr [esi+edi],ax
            shr edi,1
            loop son
    };
```

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

En dıştaki for döngüsü bitince bütün değerler hesaplanmış olur. Hesaplanan değerler stack’ten çekilip diziye teker teker eklenir ve program sona erer.

4- Erosion C++ kodu

Erosion işleminin C++ kodu aşağıdaki gibidir. Dilation işleminin neredeyse bire bir aynısıdır. Sadece 0 ile başlatılan max yerine 255 ile başlatılan min değişkeni vardır ve en içteki if kontrolünde > yerine < işareti kullanılmıştır. Farklı olan bu kısımlar işaretlenmiştir.

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

    int half_f_size = filter_size / 2;
    short* resim_copy = (short*)malloc(n * sizeof(short));

    int row_size = sqrt(n);

    // resim_org'un her pikseli için
    for (int i = 0; i < n; i++) {

        int min_pixel = 255;
        // filtre çevresindeki piksellerin kontrolü
        for (int j = -half_f_size; j <= half_f_size; j++) {
            for (int k = -half_f_size; k <= half_f_size; k++) {
                // resmin sınırlarını aşmaması için
                if ((i + j * row_size + k >= 0) && (i + j * row_size + k < n)) {
                    // en küçük pikselin bulunması
                    if (resim_org[i + j * row_size + k] < min_pixel) {
                        min_pixel = resim_org[i + j * row_size + k];
                    }
                }
            }
        }

        resim_copy[i] = min_pixel;
    }

    // resim_copy --> resim_org kopyalama
    for (int i = 0; i < n; i++) {
        resim_org[i] = resim_copy[i];
    }

    free(resim_copy);
    printf("\nErosion islemi sonucunda resim \"eroded.pgm\" ismiyle
olusturuldu...\n");
}
```

5- Erosion assembly kodu

Erosion işlemi dilation işlemine çok benzediği için sadece farklı olan 2 satır işaretlenmiştir.

```
void Erosion(int n, int filter_size, short* resim_org) {
    __asm {
            //resmin kare oldugu bilindigi icin satir veya sutun sayisi n'nin
            //karekoku hesaplanarak bulunur. islem sonunda ecx = satir sayisi
            mov ebx, n
            mov ecx, 0
            mov edx, 0
        iter:
            inc ecx
            mov eax, ecx
            mul ecx
            cmp eax, ebx
            jge done
            jmp iter
            done : ;//ecx = sqrt(n) = satir sayisi

            mov esi, 0; // esi = i

        start2:
            cmp esi, n; // i == n
            je exit2; // dongu sona erer
            mov edx, filter_size
            shr edx, 1;// filter_size/2
            mov ebx, edx
            neg ebx;// ebx = j = -filter_size/2
            mov eax, 255; // eax = min

        start3:
            mov edx, filter_size;
            shr edx, 1;// filter_size/2
            cmp ebx, edx; // j == filter_size/2
            jg exit3; // dongu sona erer
            mov edi, edx
            neg edi; // edi = k = -filter_size/2

        start4:
            mov edx, filter_size
            shr edx, 1;// filter_size/2
            cmp edi, edx; // k == filter_size/2
            jg exit4; // dongu sona erer

            push ebx;// j degerini korumak icin push islemi
            push eax;// min degerini korumak icin push islemi
            mov eax, ecx;// eax = satir sayisi
```

```
            xor edx, edx
            mul ebx;// eax = j * satir sayisi
            add eax, esi;// eax = i + j * satir sayisi
            add eax, edi; // eax = i + j * satir sayisi + k

            cmp eax, 0;// Eger eax 0'dan kucukse dizi disina cikildigindan devam
edilmez
            jl exit5
            cmp eax, n; // Eger eax n'ye buyuk esitse dizi disina cikildigindan
devam edilmez
            jge exit5

            pop ebx;// stack'e en son atilan min degeri ebx'e cekilir ama stack'de
de durmasi istenildiginden tekrar push yapilir
            push ebx
            xchg eax, ebx;// eax = min , ebx = i + j * satir sayisi + k
            push esi;// i degerini korumak icin push islemi
            mov esi, resim_org
            shl ebx, 1
            cmp word ptr[esi + ebx], ax;// resim_org[i + j * satir sayisi + k] ==
min
            jge exit6;//eger dizide kontrol edilen deger min'den buyuk esitse devam
edilmez
            mov ax, word ptr[esi + ebx];// yeni min degeri ax'e alinir
        exit6:
            shr ebx, 1
            pop esi
            pop edx// cop deger
            pop ebx
            inc edi;// k++
            jmp start4
        exit5 :
            pop eax
            pop ebx
            inc edi; // k++
            jmp start4; // basa don
        exit4:
            inc ebx; // j++
            jmp start3; // basa don
        exit3:
            push ax // max elemani bulma islemi bitti, max deger stack'e atilir
            inc esi; // i++
            jmp start2; // basa don
        exit2:
            mov ecx, n // tum donguler sona erdi
            mov edi, n // stackten max degerler teker teker alinip diziye sondan
basa dogru eklenir
            mov esi, resim_org
        son :
            dec edi
            pop ax
```

```
            shl edi, 1
            mov word ptr[esi + edi], ax
            shr edi, 1
            loop son
    };
    printf("\nErosion islemi sonucunda resim \"eroded.pgm\" ismiyle
olusturuldu...\n");
}
```