GÖRÜNTÜ İŞLEME

Gri resimlere Kenar Belirleme (Edge Detection) işlemi uygulanması

CEREN
YAŞAR
14011020

Yöntem Bölümü

Kenar belirleme yöntemini yaparken elimdeki 3*3 lük filtreleri gerekli sırada çarparak Gx ve Gy değerlerini elde ettim ve bu değerlerin mutlak değerlerini aldım. Sonraki aşamada normalizasyon işlemi uyguladım ve en sonda matrisin dışına dışındaki çerçevenin pixel değerlerini kopyaladım.

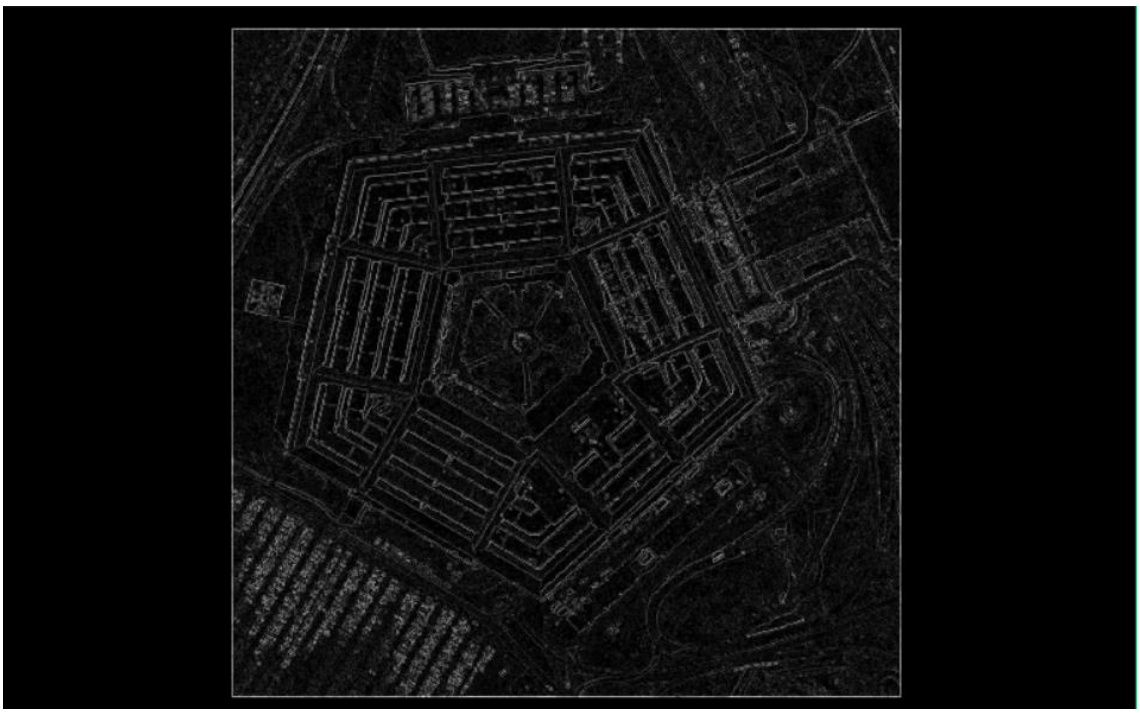
Eşik belirleme yönteminde ise dosya da belirtilen aşamaları gerçekleyen bir algoritma tasarladım.

Siyah-beyaz seçeneğini elde etmek için de sobel+global eşik seviyesi yöntemlerini uyguladım.

Uygulama Bölümü

1- pentagone.1024

1- Sobel yöntemi



2-Prewitt Yöntemi

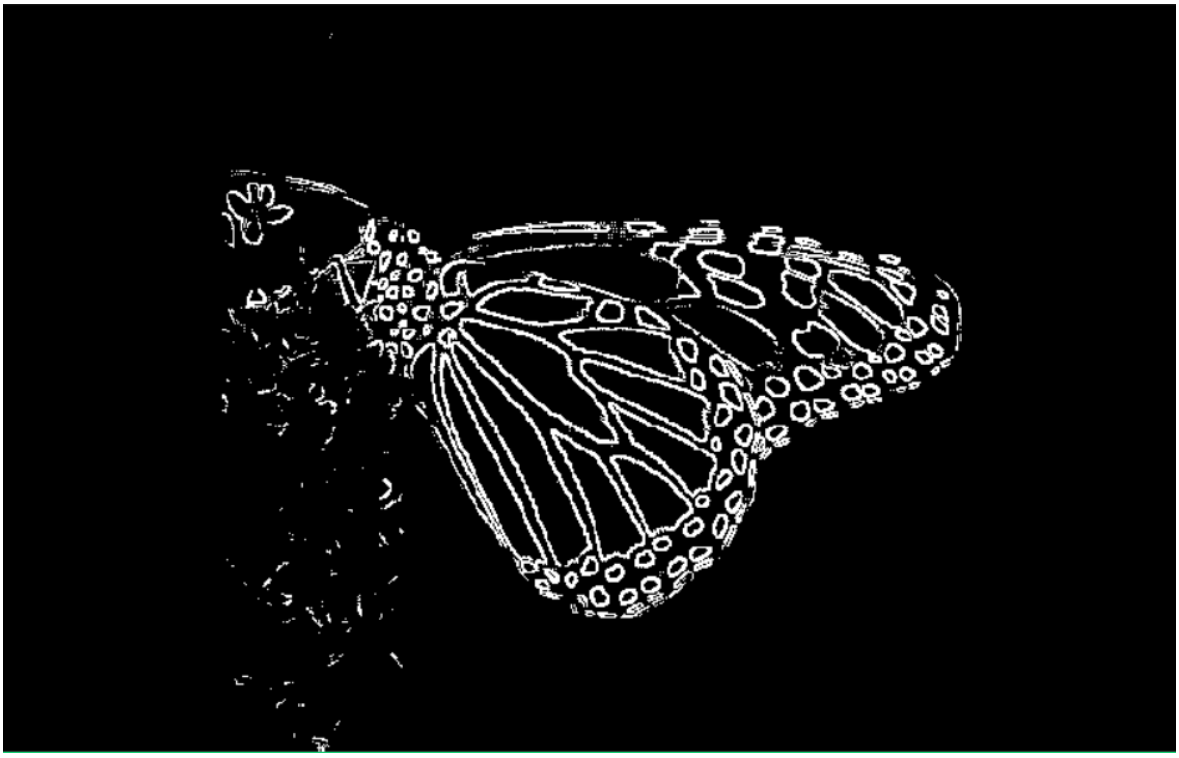
3-Siyah-Beyaz

2- monarch.512

1- Sobel Yöntemi

2- Prewitt Yöntemi

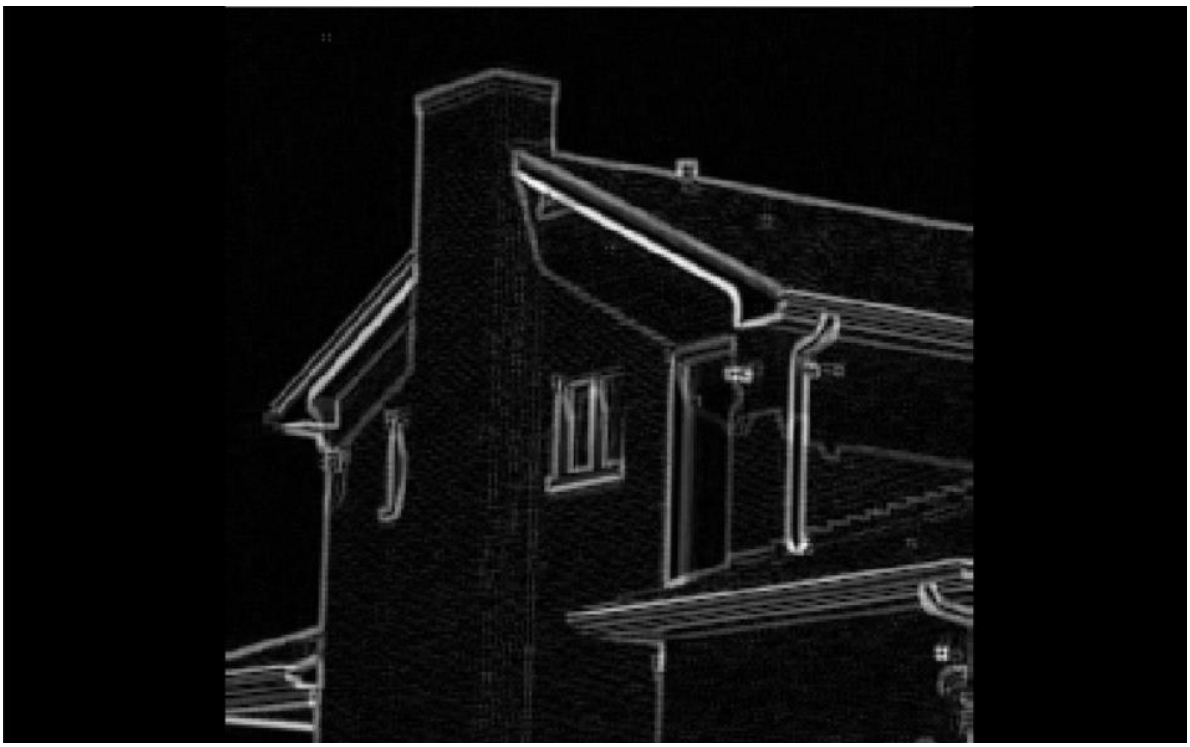
3- Siyah-Beyaz



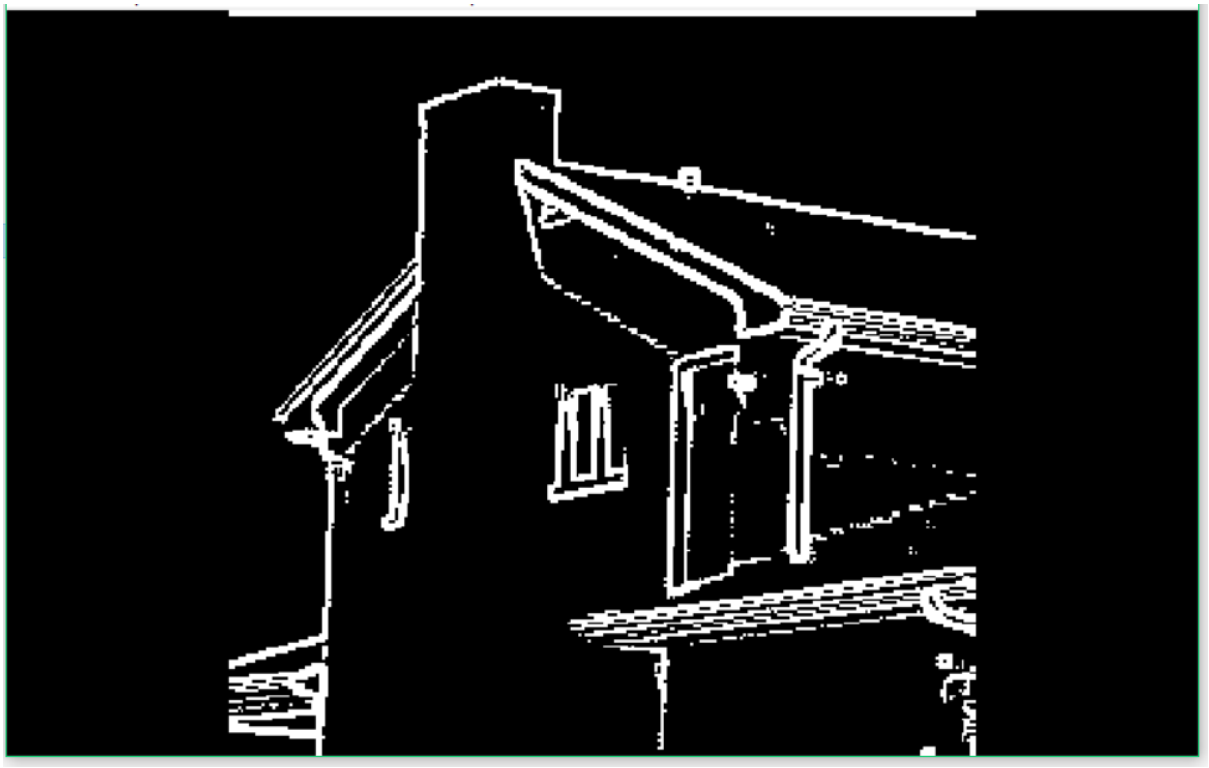
3-house.256

1-Sobel Yöntemi

3- Prewitt Yöntemi



3-Siyah-Beyaz



Sonuç Bölümü

Sobel ve Prewitt filtreleri birbirine çok yakın sonuçlar çıkarıyor. Kenar belirtmede bir miktar eksiklikleri var. Sobel kenarları belirtme konusunda küçük bir miktar da olsa daha iyi sonuç veriyor.Çok net bir şekilde kenar belirtmek için global eşik yöntemi ile kullanırsak kenar belirleme işlevinde daha iyi iş çıkarıyor.

Global eşik yöntemi direk resme uygulandığında örneğin bizim house resmimizde evin dışında kalan gökyüzü alanını da kenar sanıyor ve beyaz gösteriyor. Bu noktada sobel ya da prewitt yardımıyla global eşiği kullandığımızda istediğimiz kenar beyaz gerisi siyah olan resmi elde edebiliyoruz.

KOD

```
import sys
from math import sqrt

fileName="house.256.pgm"
try:
    fout=open(fileName,"rb")
except:
    print ("Cannot open file ", filename, "Exiting … \n")
    sys.exit()
#read function
def read_pgm(f):
    f.readline()
    size=f.readline()
    width =int(size.split()[0])
    height=int(size.split()[1])
    depth=int(f.readline())
    matrix=[]
    for y in range(height):
        row = []
        for y in range(width):
            row.append(ord(f.read(1)))
        matrix.append(row)
    f.close()
    return [matrix, width, height, depth]


#write function
def write_pgm(matrix, width, height, depth, filename):
   try:
        fout=open(filename, 'wb')
   except:
        print ("Cannot open file ", filename, "Exiting … \n")
        sys.exit()
   #define header
   pgmHeader='P5'+'\n'+str(width)+' '+str(height)+'\n'+str(depth)+"\n"
   fout.write(str.encode(pgmHeader))
   for i in matrix:
       fout.write(bytes(i))
   fout.close()

#photo's information
arr=read_pgm(fout)
matrix=arr[0]
width=arr[1]
height=arr[2]
depth=arr[3]

fout.close()
#define filter
p_x=[[-1,0,1],[-1,0,1],[-1,0,1]]
p_y=[[1,1,1],[0,0,0],[-1,-1,-1]]
s_x=[[-1,0,1],[-2,0,2],[-1,0,1]]
s_y=[[1,2,1],[0,0,0],[-1,-2,-1]]

def convolution(filterX,filterY,matrix,width,height,depth):
    newMat=[]
    for i in range(1,height-1):
        row=[]
```

```
        for j in range(1,width-1):
            gx=0
            gy=0
            gx=abs((matrix[i+1][j]*filterX[2][1]+matrix[i-
1][j]*filterX[0][1]) + (matrix[i+1][j-1]*filterX[2][0]+matrix[i-1][j-
1]*filterX[0][0])+(matrix[i+1][j+1]*filterX[2][2]+matrix[i-
1][j+1]*filterX[0][2]))
            gy=abs((matrix[i][j+1]*filterY[1][2]+matrix[i][j-
1]*filterY[1][0]) + (matrix[i-1][j+1]*filterY[0][2]+matrix[i-1][j-
1]*filterY[0][0])+(matrix[i+1][j+1]*filterY[2][2]+matrix[i+1][j-
1]*filterY[2][0]))
            row.append(int(gx+gy))
        newMat.append(row)
    #find max pixel
    maxNew=newMat[0][0]
    for i in range(0,width-2):
        for j in range(0,height-2):
            if newMat[i][j] > maxNew:
                maxNew=newMat[i][j]
    #normalization
    if maxNew > 255 :
        for i in range(0,width-2):
            for j in range(0,height-2):
                newMat[i][j]=int(newMat[i][j]*255/maxNew)
    for i in newMat:
        first=i[0]
        last=i[-1]
        i.insert(0,first)
        i.append(last)
    firstLine=newMat[0]
    lastLine=newMat[-1]
    newMat.insert(0,firstLine)
    newMat.append(lastLine)
    return newMat
# global
def globalEsik(T,matrix):
    T2=0
    while abs(T-T2)>2:
        g1=[]
        g2=[]
        for i in matrix:
            for j in i:
                if j > T :
                    g2.append(j)
                else:
                    g1.append(j)
        m1=0
        m2=0
        for i in g1:
            m1+=i
        m1=int(m1/len(g1))
        for j in g2:
            m2+=j
        m2=int(m2/len(g2))
        T2=T
        T=int((m1+m2)/2)
    new_Mat=[]
    for i in matrix:
        row=[]
        for j in i:
            if j > T:
```

```
                row.append(255)
            else:
                row.append(0)
        new_Mat.append(row)
    return new_Mat



cntr=True
while cntr:
    print (fileName + " dosyasına uygulamak istediğiniz işlemi seçin.\n")
    option=input("1-Sobel Kenar tanıma filtresi\n2-Prewitt Kenar tanıma
filtesi\n3-Siyah-beyaz filtre\n4-Çıkış")
    if option=='1':
        print("İşlem gerçekleştiriliyor...")
        newFile="Sobel_"+fileName
        new_Mat=convolution(s_x,s_y,matrix,width,height,depth)
        write_pgm(new_Mat,width,height,depth,newFile)
        print("İşlem gerçekleştirildi. Sonuç için başlangıç resminizin
olduğu dizine bakın.")
    elif option=='2':
        print("İşlem gerçekleştiriliyor...")
        newFile="Prewitt_"+fileName
        new_Mat=convolution(p_x,p_y,matrix,width,height,depth)
        write_pgm(new_Mat,width,height,depth,newFile)
        print("İşlem gerçekleştirildi. Sonuç için başlangıç resminizin
olduğu dizine bakın.")
    elif option=='3':
        T=input("T değerini seçin.")
        print("İşlem gerçekleştiriliyor...")
        newFile="SiyahBeyaz_"+fileName
        new_Mat=convolution(s_x,s_y,matrix,width,height,depth)
        new_Mat2=globalEsik(int(T),new_Mat)
        write_pgm(new_Mat2,width,height,depth,newFile)
        print("İşlem gerçekleştirildi. Sonuç için başlangıç resminizin
olduğu dizine bakın.")
    elif option=='4':
        cntr=False
    else :
        print("Yanlış seçim")
```