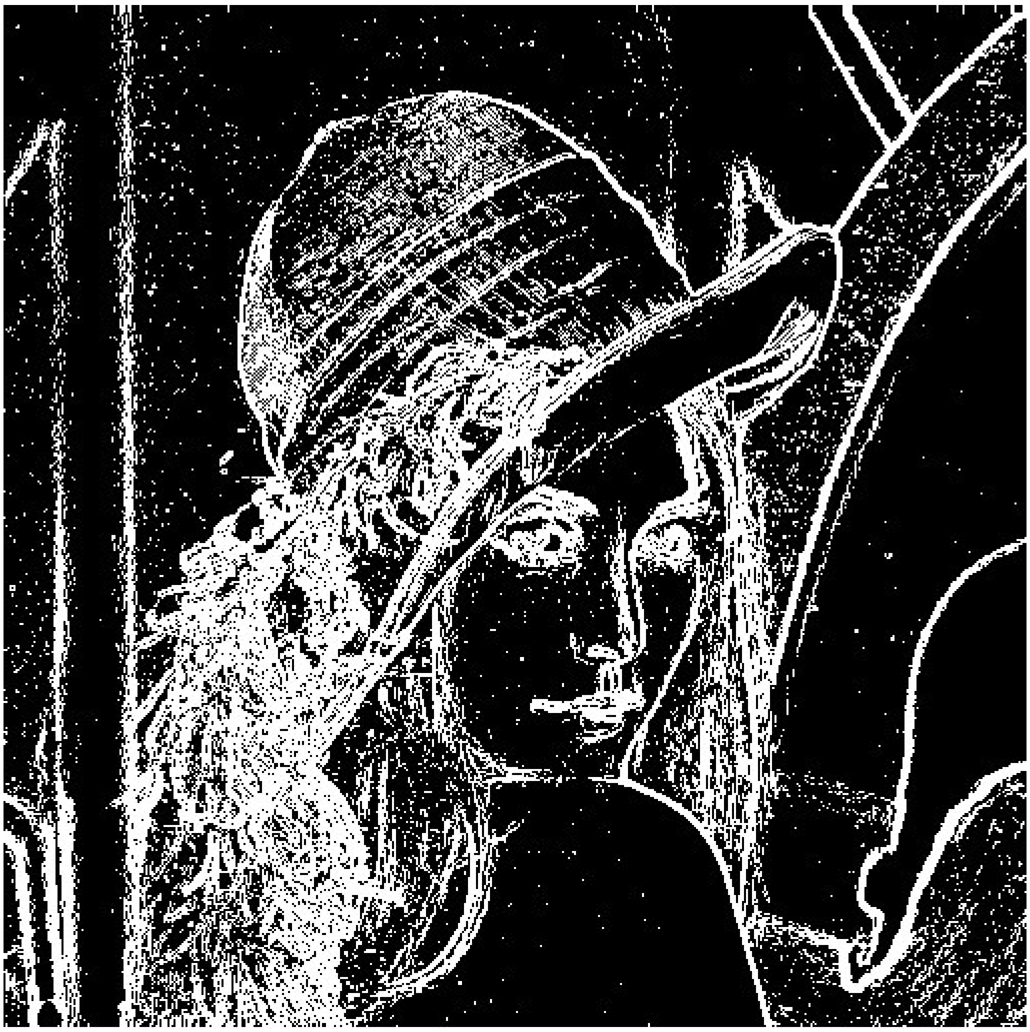
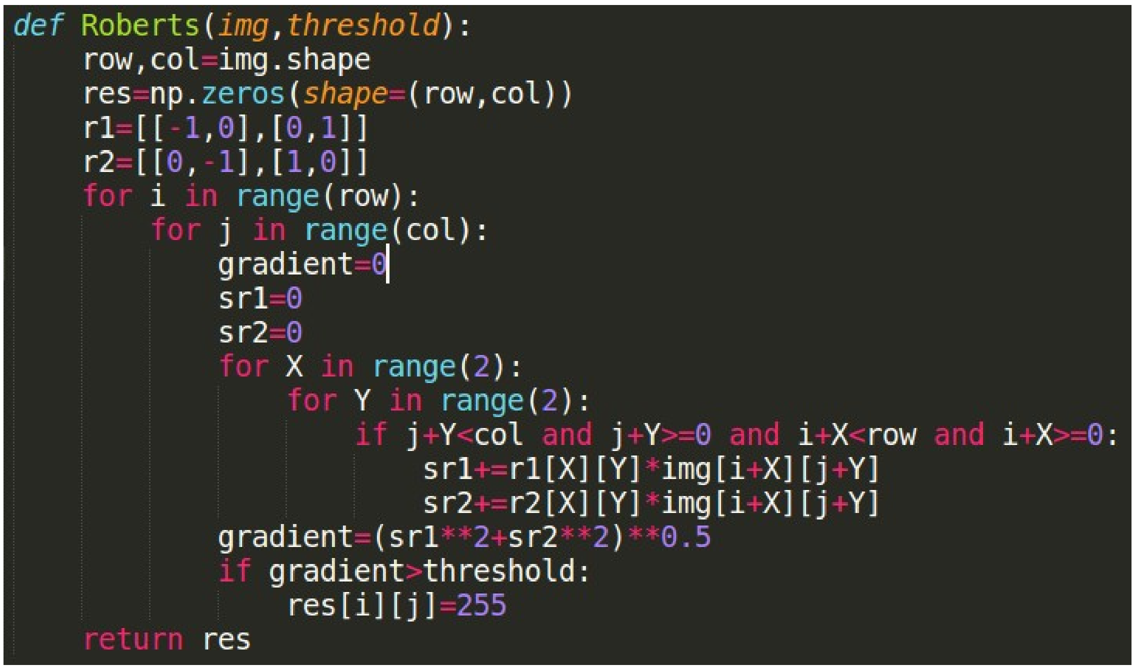
Computer Vision hw9 Report

R07921010 余奇安

1.Robert's operator, threshold at 12

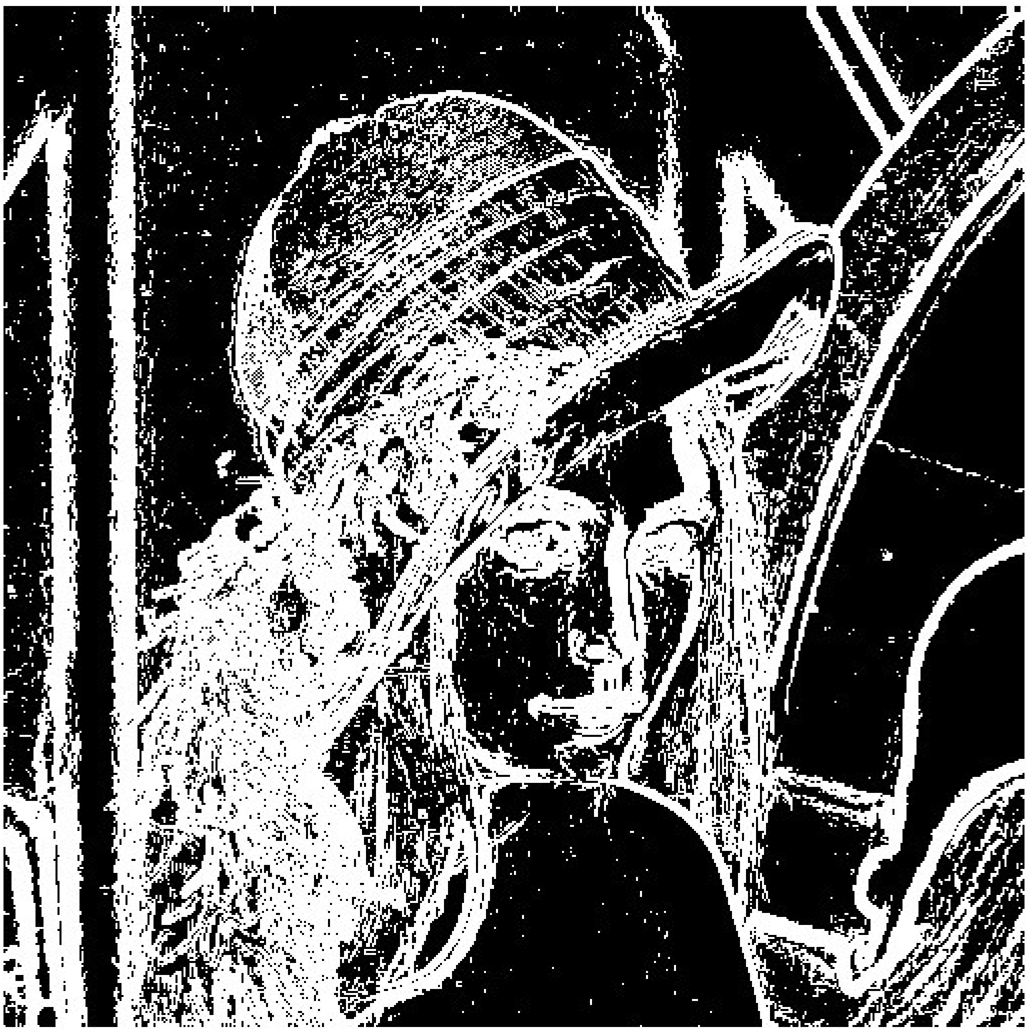


Code:

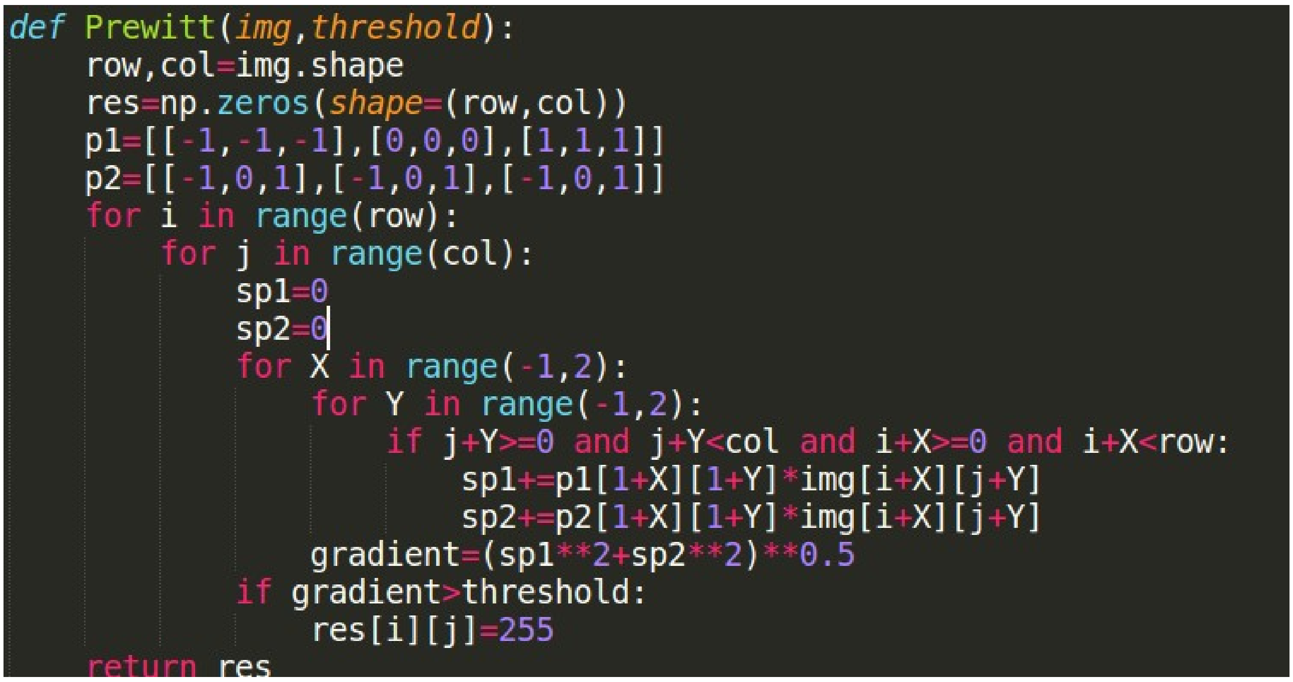


```
def Roberts(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    r1=[[-1,0],[0,1]]
    r2=[[0,-1],[1,0]]
    for i in range(row):
        for j in range(col):
            gradient=0
            sr1=0
            sr2=0
            for X in range(2):
                for Y in range(2):
                    if j+Y<col and j+Y>=0 and i+X<row and i+X>=0:
                        sr1+=r1[X][Y]*img[i+X][j+Y]
                        sr2+=r2[X][Y]*img[i+X][j+Y]
            gradient=(sr1**2+sr2**2)**0.5
            if gradient>threshold:
                res[i][j]=255
    return res
```

2.Prewitt’s edge detector threshold at 24

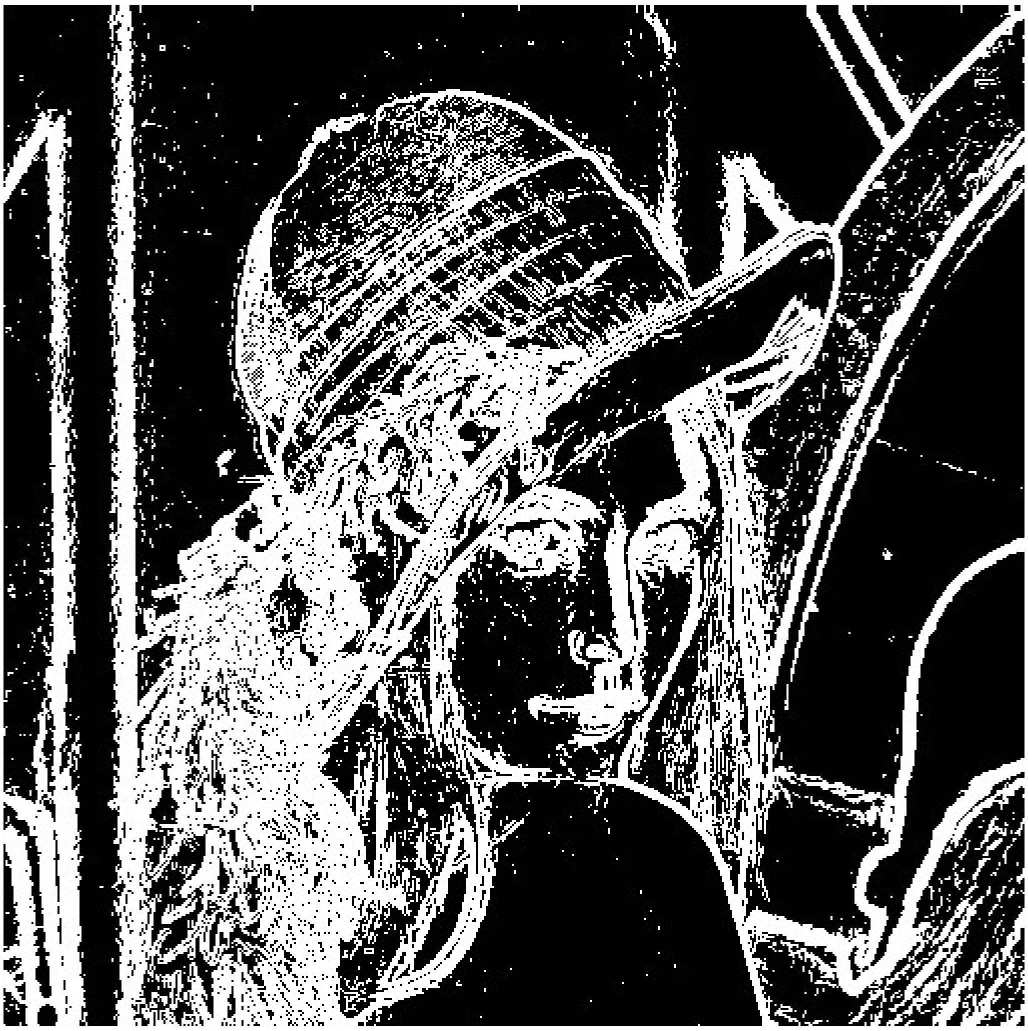


Code:

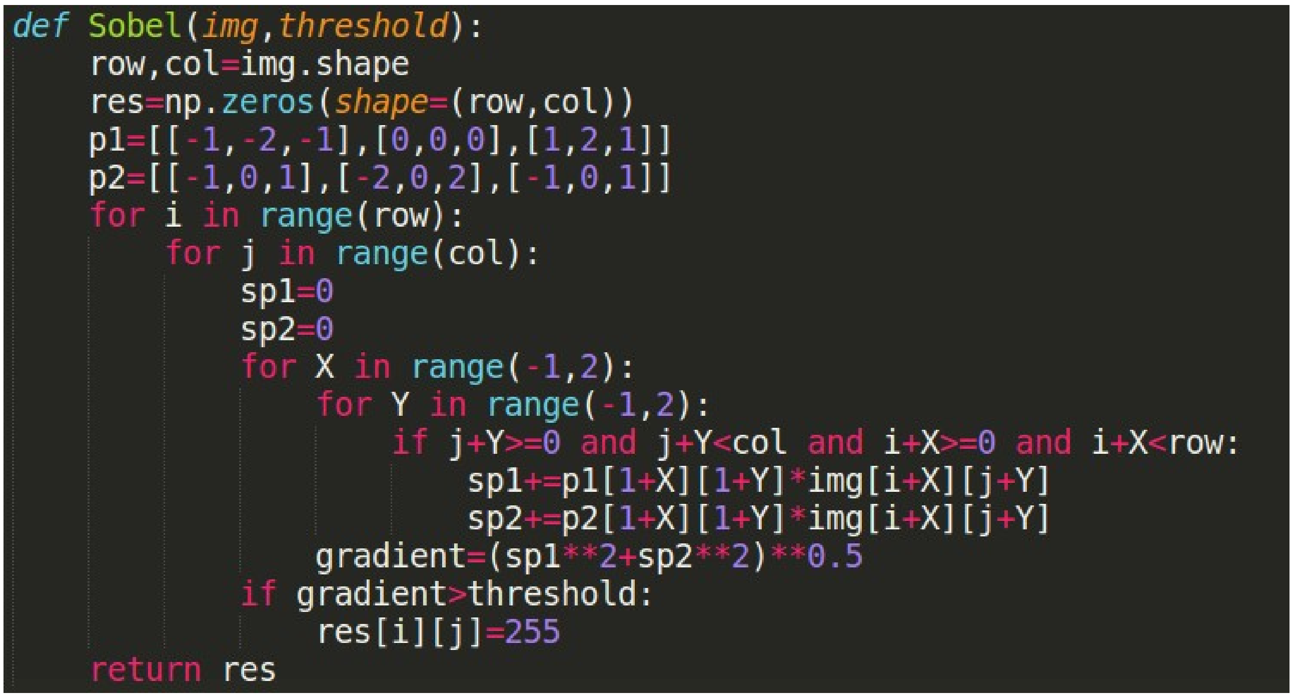


```python
def Prewitt(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    p1=[[-1,-1,-1],[0,0,0],[1,1,1]]
    p2=[[-1,0,1],[-1,0,1],[-1,0,1]]
    for i in range(row):
        for j in range(col):
            sp1=0
            sp2=0
            for X in range(-1,2):
                for Y in range(-1,2):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                        sp1+=p1[1+X][1+Y]*img[i+X][j+Y]
                        sp2+=p2[1+X][1+Y]*img[i+X][j+Y]
                gradient=(sp1**2+sp2**2)**0.5
            if gradient>threshold:
                res[i][j]=255
    return res
```

3.Sobel's Edge Detector at 38

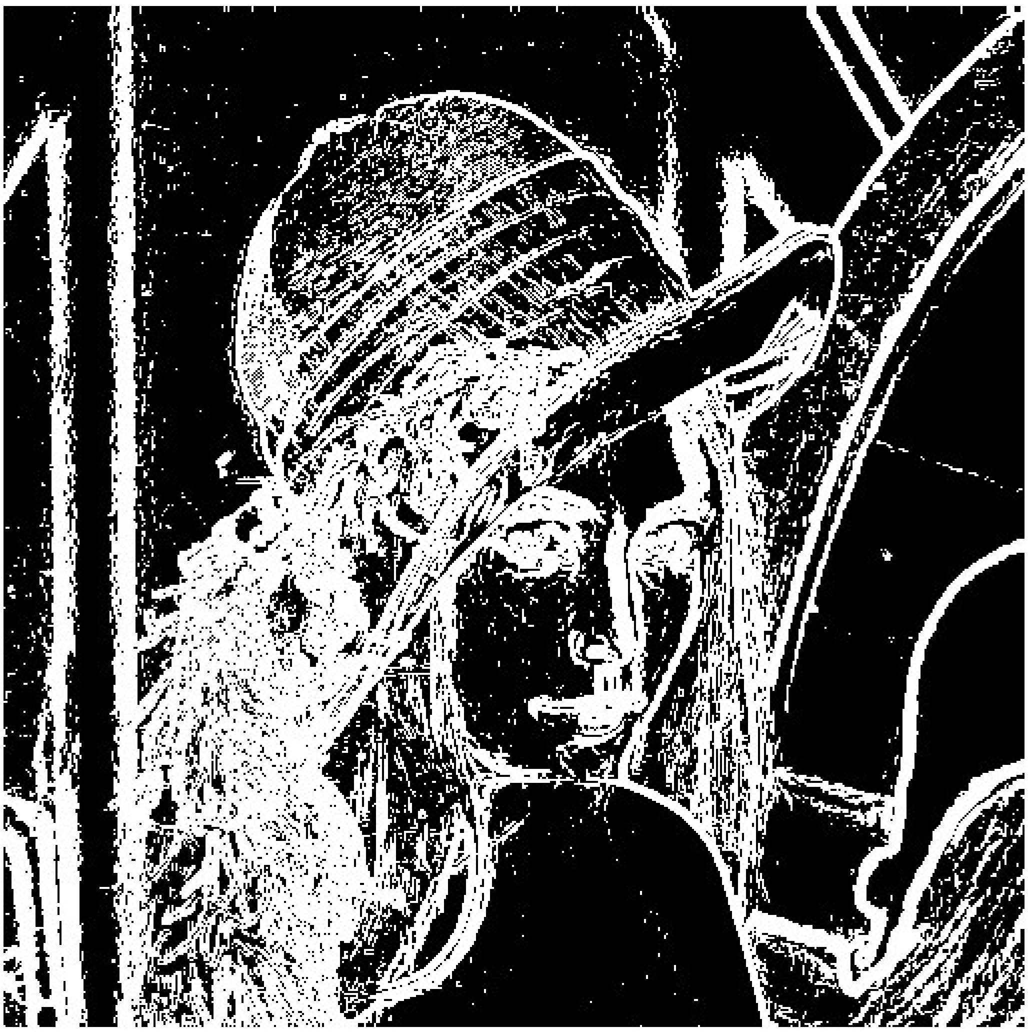


Code:

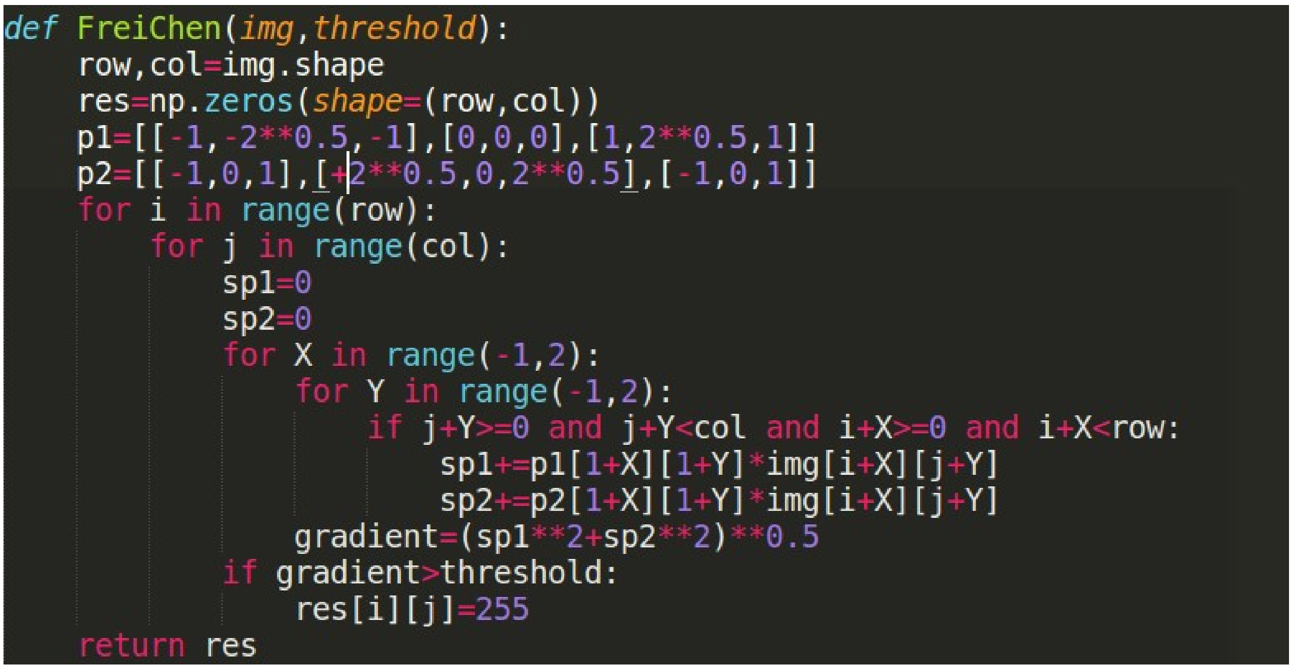


```
def Sobel(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    p1=[[-1,-2,-1],[0,0,0],[1,2,1]]
    p2=[[-1,0,1],[-2,0,2],[-1,0,1]]
    for i in range(row):
        for j in range(col):
            sp1=0
            sp2=0
            for X in range(-1,2):
                for Y in range(-1,2):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                        sp1+=p1[1+X][1+Y]*img[i+X][j+Y]
                        sp2+=p2[1+X][1+Y]*img[i+X][j+Y]
                gradient=(sp1**2+sp2**2)**0.5
            if gradient>threshold:
                res[i][j]=255
    return res
```

4.Frei and Chen’s Gradient Operator at 30

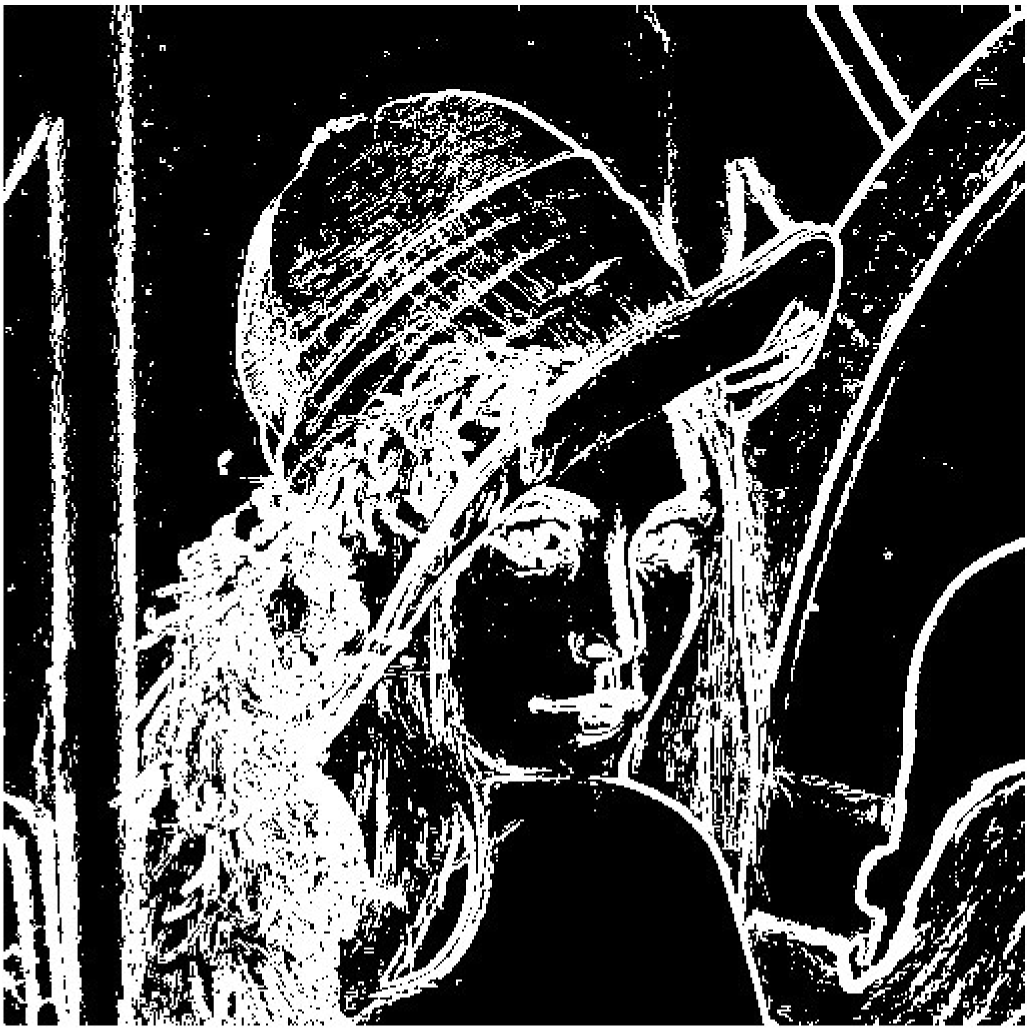


Code:

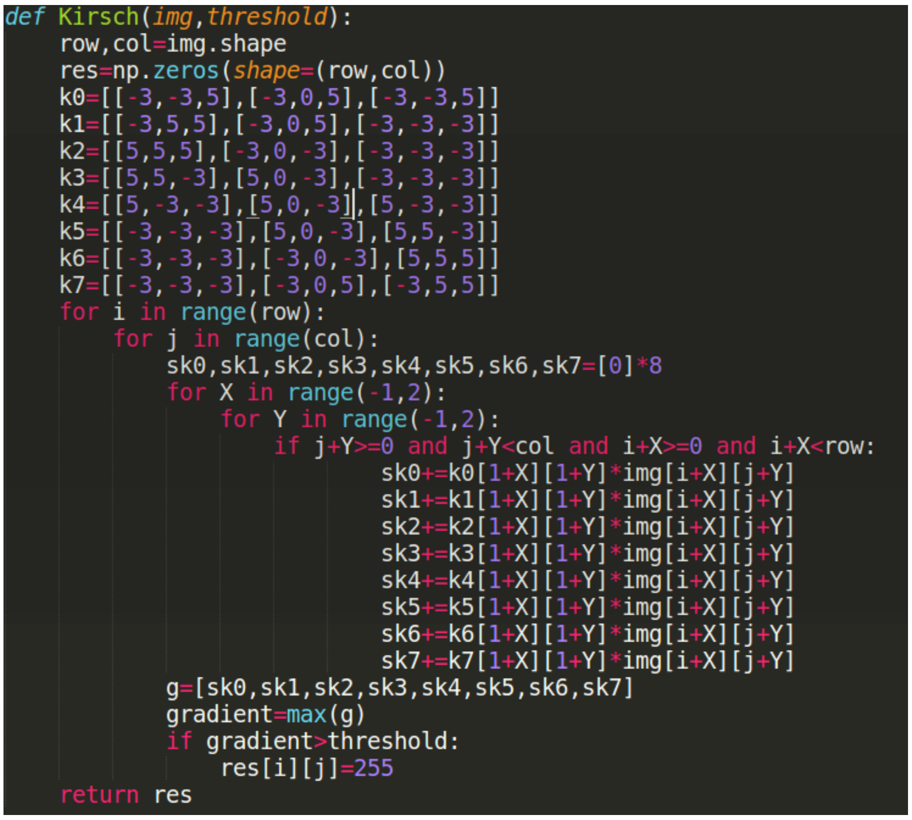


```python
def FreiChen(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    p1=[[-1,-2**0.5,-1],[0,0,0],[1,2**0.5,1]]
    p2=[[-1,0,1],[-2**0.5,0,2**0.5],[-1,0,1]]
    for i in range(row):
        for j in range(col):
            sp1=0
            sp2=0
            for X in range(-1,2):
                for Y in range(-1,2):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                        sp1+=p1[1+X][1+Y]*img[i+X][j+Y]
                        sp2+=p2[1+X][1+Y]*img[i+X][j+Y]
                gradient=(sp1**2+sp2**2)**0.5
            if gradient>threshold:
                res[i][j]=255
    return res
```

5.Kirsch’s Compass Operator at 135

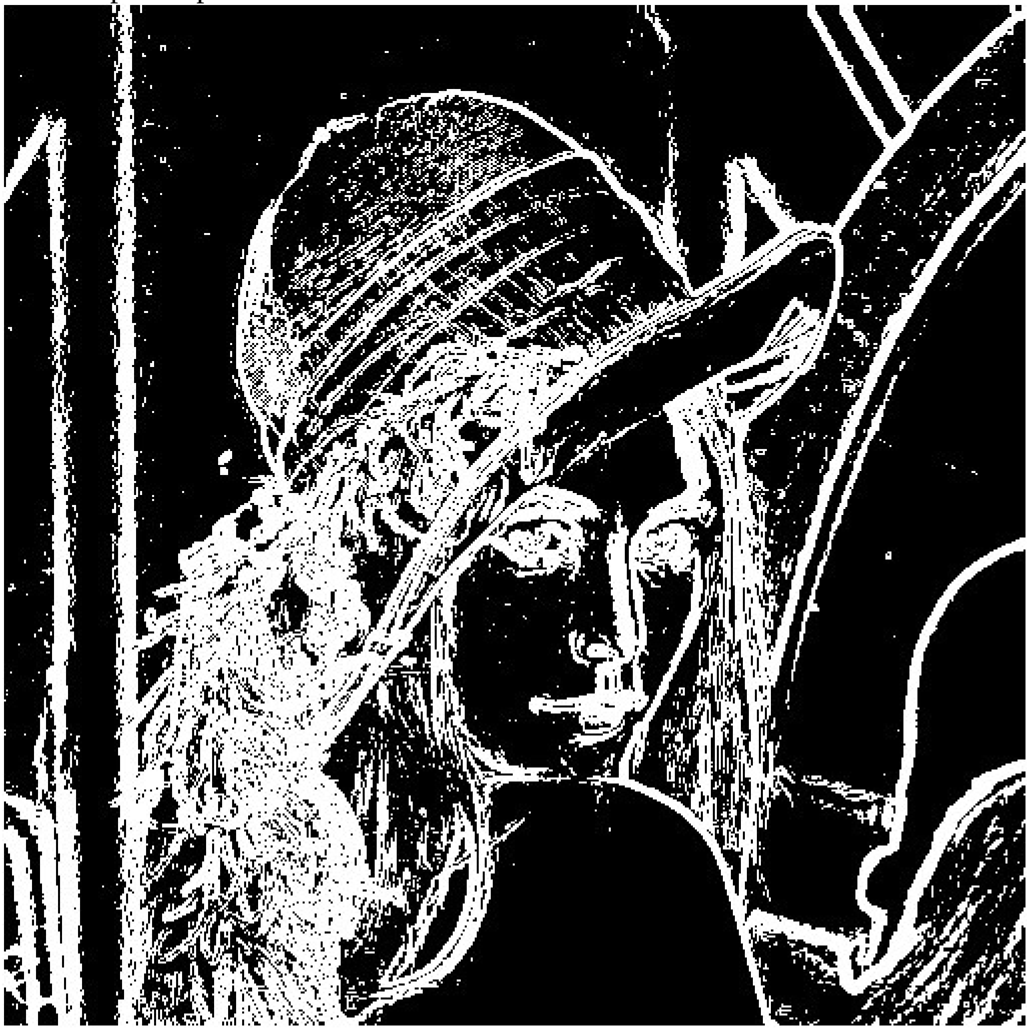


Code:

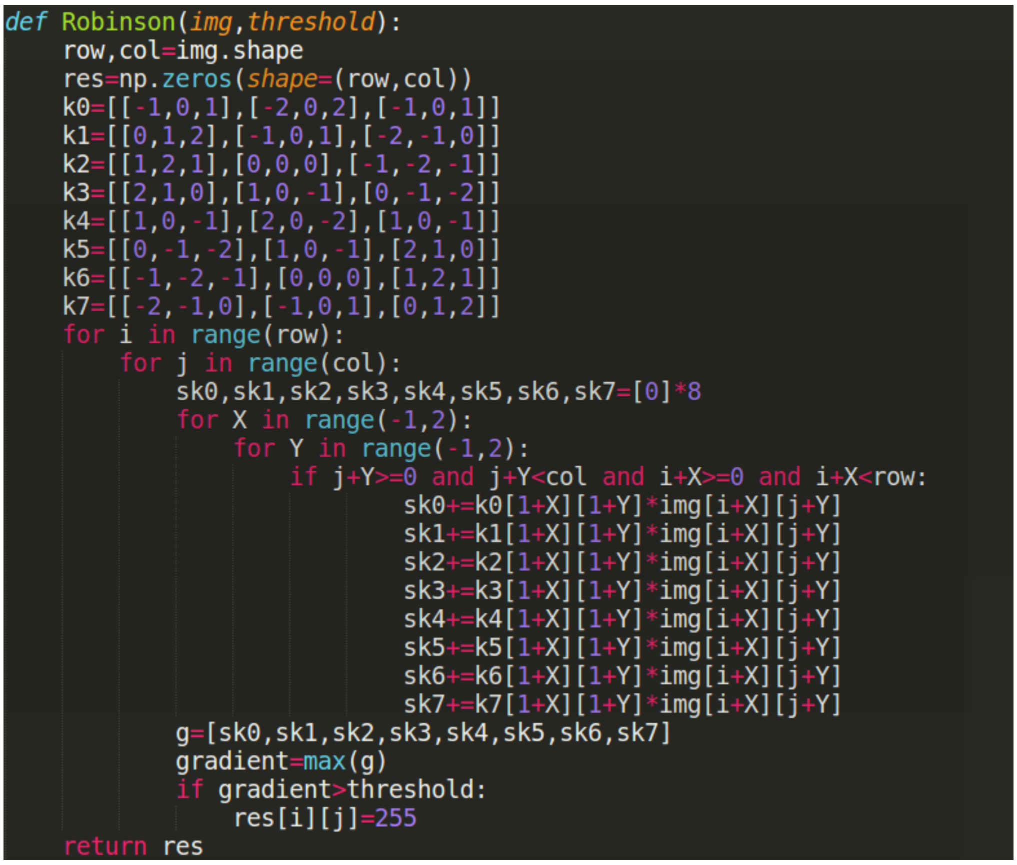


```
def Kirsch(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    k0=[[-3,-3,5],[-3,0,5],[-3,-3,5]]
    k1=[[-3,5,5],[-3,0,5],[-3,-3,-3]]
    k2=[[5,5,5],[-3,0,-3],[-3,-3,-3]]
    k3=[[5,5,-3],[5,0,-3],[-3,-3,-3]]
    k4=[[5,-3,-3],[5,0,-3],[5,-3,-3]]
    k5=[[-3,-3,-3],[5,0,-3],[5,5,-3]]
    k6=[[-3,-3,-3],[-3,0,-3],[5,5,5]]
    k7=[[-3,-3,-3],[-3,0,5],[-3,5,5]]
    for i in range(row):
        for j in range(col):
            sk0,sk1,sk2,sk3,sk4,sk5,sk6,sk7=[0]*8
            for X in range(-1,2):
                for Y in range(-1,2):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                            sk0+=k0[1+X][1+Y]*img[i+X][j+Y]
                            sk1+=k1[1+X][1+Y]*img[i+X][j+Y]
                            sk2+=k2[1+X][1+Y]*img[i+X][j+Y]
                            sk3+=k3[1+X][1+Y]*img[i+X][j+Y]
                            sk4+=k4[1+X][1+Y]*img[i+X][j+Y]
                            sk5+=k5[1+X][1+Y]*img[i+X][j+Y]
                            sk6+=k6[1+X][1+Y]*img[i+X][j+Y]
                            sk7+=k7[1+X][1+Y]*img[i+X][j+Y]
            g=[sk0,sk1,sk2,sk3,sk4,sk5,sk6,sk7]
            gradient=max(g)
            if gradient>threshold:
                res[i][j]=255
    return res
```

6.Robinson’s Compass Operator at 43

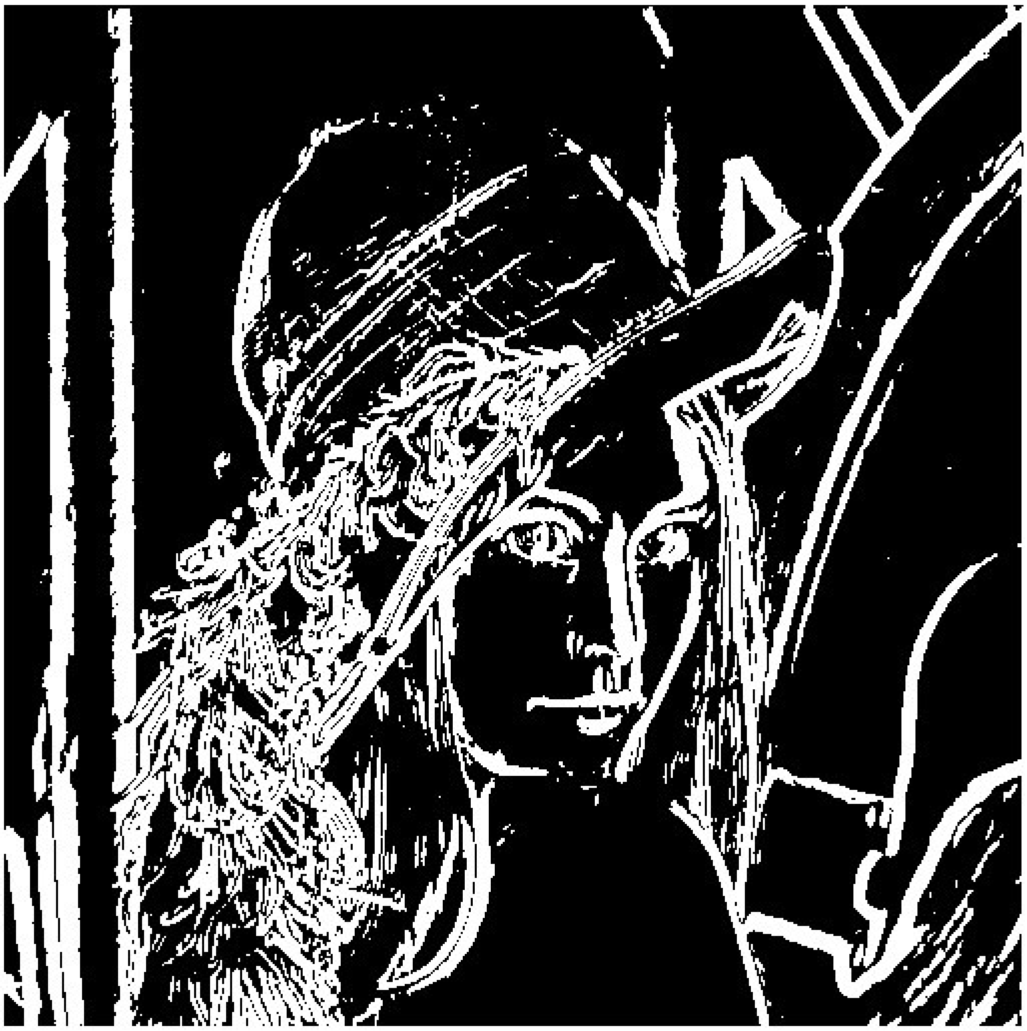


Code:

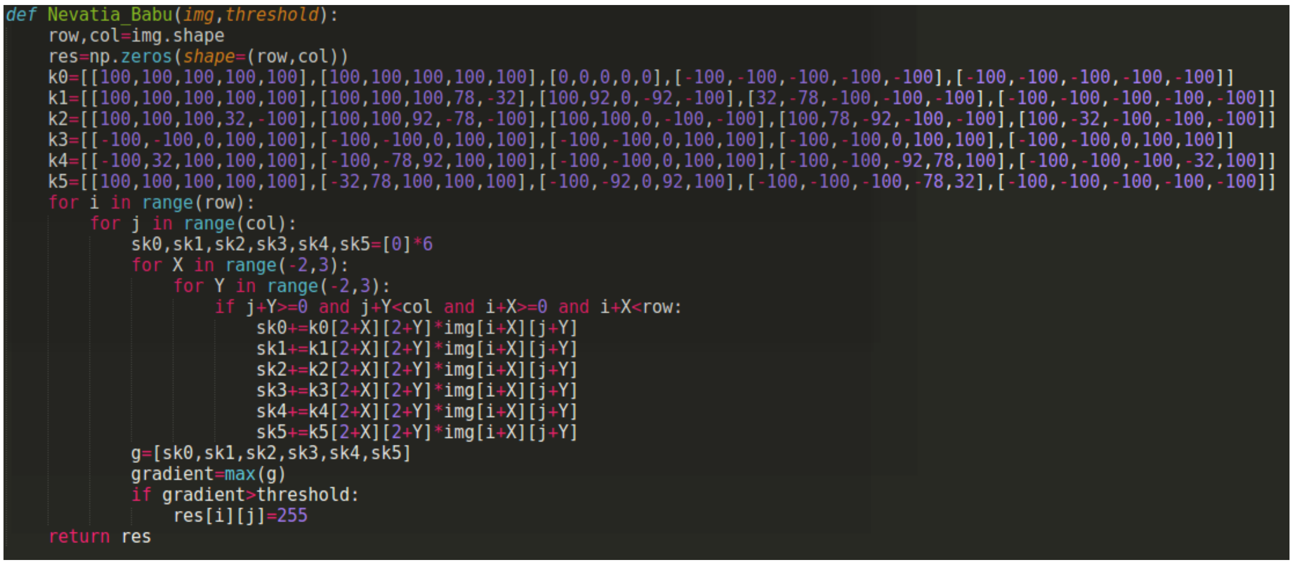


```
def Robinson(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    k0=[[-1,0,1],[-2,0,2],[-1,0,1]]
    k1=[[0,1,2],[-1,0,1],[-2,-1,0]]
    k2=[[1,2,1],[0,0,0],[-1,-2,-1]]
    k3=[[2,1,0],[1,0,-1],[0,-1,-2]]
    k4=[[1,0,-1],[2,0,-2],[1,0,-1]]
    k5=[[0,-1,-2],[1,0,-1],[2,1,0]]
    k6=[[-1,-2,-1],[0,0,0],[1,2,1]]
    k7=[[-2,-1,0],[-1,0,1],[0,1,2]]
    for i in range(row):
        for j in range(col):
            sk0,sk1,sk2,sk3,sk4,sk5,sk6,sk7=[0]*8
            for X in range(-1,2):
                for Y in range(-1,2):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                            sk0+=k0[1+X][1+Y]*img[i+X][j+Y]
                            sk1+=k1[1+X][1+Y]*img[i+X][j+Y]
                            sk2+=k2[1+X][1+Y]*img[i+X][j+Y]
                            sk3+=k3[1+X][1+Y]*img[i+X][j+Y]
                            sk4+=k4[1+X][1+Y]*img[i+X][j+Y]
                            sk5+=k5[1+X][1+Y]*img[i+X][j+Y]
                            sk6+=k6[1+X][1+Y]*img[i+X][j+Y]
                            sk7+=k7[1+X][1+Y]*img[i+X][j+Y]
            g=[sk0,sk1,sk2,sk3,sk4,sk5,sk6,sk7]
            gradient=max(g)
            if gradient>threshold:
                res[i][j]=255
    return res
```

7. Nevatia-Babu 5x5 operator at 12500



Code:



```
def Nevatia_Babu(img,threshold):
    row,col=img.shape
    res=np.zeros(shape=(row,col))
    k0=[[100,100,100,100,100],[100,100,100,100,100],[0,0,0,0,0],[-100,-100,-100,-100,-100],[-100,-100,-100,-100,-100]]
    k1=[[100,100,100,100,100],[100,100,100,78,-32],[100,92,0,-92,-100],[32,-78,-100,-100,-100],[-100,-100,-100,-100,-100]]
    k2=[[100,100,100,32,-100],[100,100,92,-78,-100],[100,100,0,-100,-100],[100,78,-92,-100,-100],[100,-32,-100,-100,-100]]
    k3=[[-100,-100,0,100,100],[-100,-100,0,100,100],[-100,-100,0,100,100],[-100,-100,0,100,100],[-100,-100,0,100,100]]
    k4=[[-100,32,100,100,100],[-100,-78,92,100,100],[-100,-100,0,100,100],[-100,-100,-92,78,100],[-100,-100,-100,-32,100]]
    k5=[[100,100,100,100,100],[-32,78,100,100,100],[-100,-92,0,92,100],[-100,-100,-100,-78,32],[-100,-100,-100,-100,-100]]
    for i in range(row):
        for j in range(col):
            sk0,sk1,sk2,sk3,sk4,sk5=[0]*6
            for X in range(-2,3):
                for Y in range(-2,3):
                    if j+Y>=0 and j+Y<col and i+X>=0 and i+X<row:
                        sk0+=k0[2+X][2+Y]*img[i+X][j+Y]
                        sk1+=k1[2+X][2+Y]*img[i+X][j+Y]
                        sk2+=k2[2+X][2+Y]*img[i+X][j+Y]
                        sk3+=k3[2+X][2+Y]*img[i+X][j+Y]
                        sk4+=k4[2+X][2+Y]*img[i+X][j+Y]
                        sk5+=k5[2+X][2+Y]*img[i+X][j+Y]
            g=[sk0,sk1,sk2,sk3,sk4,sk5]
            gradient=max(g)
            if gradient>threshold:
                res[i][j]=255
    return res
```

Discussion:

We can observe that the bigger the kernel of gradient edge detector, just like Nevatia-Babu 5x5, the better of noise removal performance, and the contour lines become more smooth and bold.