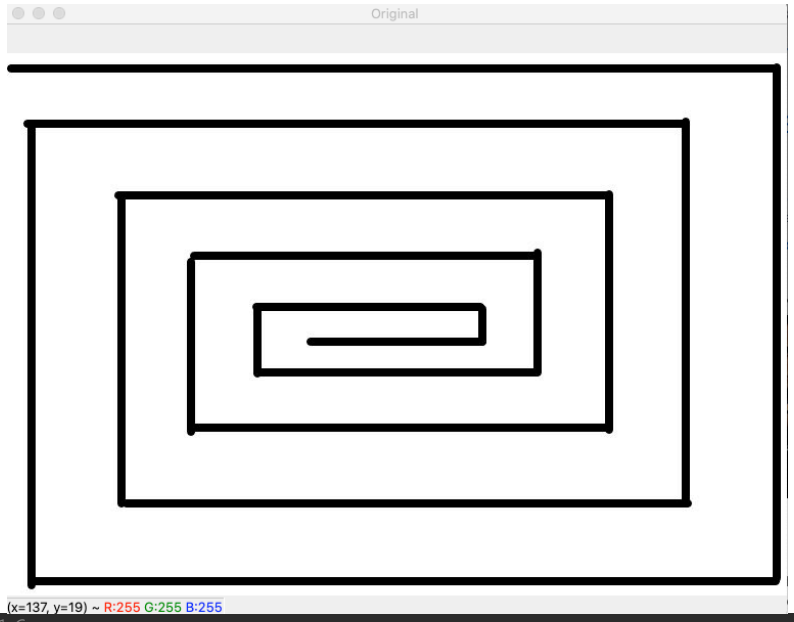
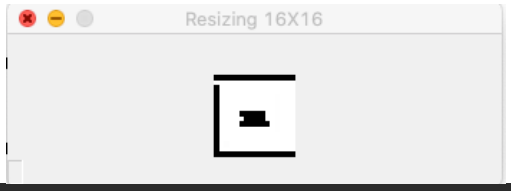
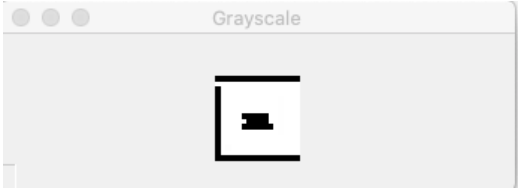
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

#1) read image
image = cv2.imread("/Users/joudabuzaid/Desktop/testImage.JPG");
cv2.imshow("Original", image)
cv2.waitKey(0)
```



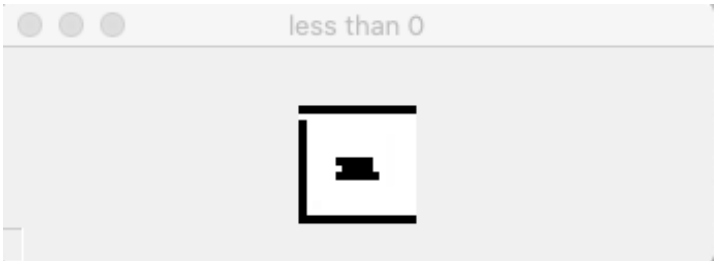

```
#2) resize 16x16
res = cv2.resize(image,(16, 16), interpolation = cv2.INTER_CUBIC)
cv2.imshow("Resizing 16X16", res)
cv2.waitKey(0)
```




```
#3) convert to gray
gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
cv2.imshow("Grayscale", gray)
cv2.waitKey(0)
```



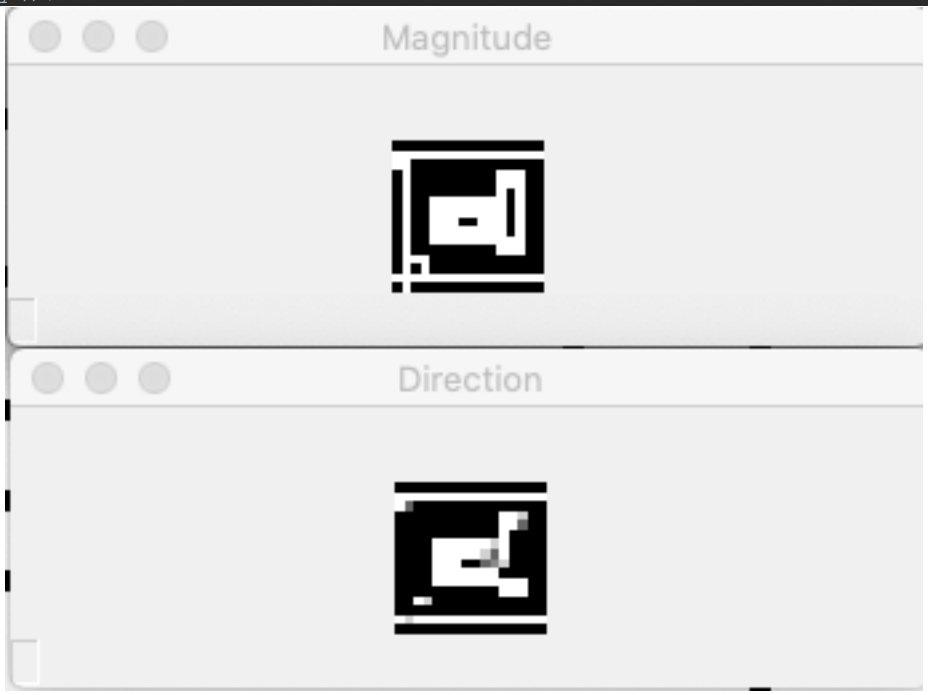
```
#4)convert to 0 if <100
Img=np.array(gray)
for i in range(16):
    for j in range(16):
        if Img[i][j]<100:
            Img[i][j]=0
cv2.imshow("less than 0", Img)
cv2.waitKey(0)
```




```
#5) matrix of pixel values
print("Img Pixel Value Matrix: ")
for i in range(16):
    for j in range(16):
        print(Img[i][j], end=" ")
    print()
```

```
Img Pixel Value Matrix:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 255 255 255 255 255 255 255 255 255 254 255 255 255
0 255 255 255 255 255 255 255 255 255 255 254 255 255 255
0 255 255 255 255 255 255 255 255 255 255 254 255 255 255
0 255 255 255 255 0 0 0 0 0 255 255 254 255 255 255
0 255 255 255 255 255 0 0 0 0 255 255 254 255 255 255
0 255 255 255 255 0 0 0 0 0 0 255 254 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 254 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 253 255 255 255 255 255 255 255 255 255 255 255 255 255
0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
#6) derivative x and y
Fy = np.array([[1.,2,1],[0,0,0],[-1,-2,-1]])
Fx = np.array([[-1,0,1.],[-2,0,2],[-1,0,1]])
Ix = cv2.filter2D(Img,cv2.CV_64F, Fx)
Iy = cv2.filter2D(Img,cv2.CV_64F, Fy)
mag1, ang = cv2.cartToPolar(Ix,Iy)
cv2.imshow("Magnitude",mag1);
cv2.imshow("Direction",ang);
cv2.waitKey();
```

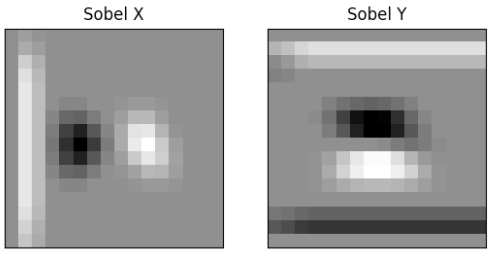



```
#7) XY pixel value matrix
print("Derivative X Matrix: ")
x=np.array(Ix)
for i in range(16):
    for j in range(16):
        print(x[i][j], end=" ")
    print()

print("Derivative Y Matrix: ")
y=np.array(Iy)
for i in range(16):
    for j in range(16):
        print(y[i][j], end=" ")
    print()
```

```
Derivative X Matrix:
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 255.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 765.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1020.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -1.0 0.0 1.0 0.0 0.0
0.0 1020.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -3.0 0.0 3.0 0.0 0.0
0.0 1020.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.0 0.0 4.0 0.0 0.0
0.0 1020.0 0.0 0.0 -255.0 -255.0 0.0 0.0 0.0 255.0 255.0 -4.0 0.0 4.0 0.0 0.0
0.0 1020.0 0.0 0.0 -510.0 -765.0 -255.0 0.0 0.0 765.0 765.0 -4.0 0.0 4.0 0.0 0.0
0.0 1020.0 0.0 0.0 -510.0 -1020.0 -510.0 0.0 0.0 765.0 1020.0 251.0 0.0 4.0 0.0 0.0
0.0 1020.0 0.0 0.0 -510.0 -765.0 -255.0 0.0 0.0 255.0 765.0 506.0 0.0 4.0 0.0 0.0
0.0 1020.0 0.0 0.0 -255.0 -255.0 0.0 0.0 0.0 0.0 255.0 252.0 0.0 3.0 0.0 0.0
0.0 1020.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -1.0 0.0 1.0 0.0 0.0
0.0 1018.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1016.0 0.0 4.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 763.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 510.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Derivative Y Matrix:
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-510.0 -765.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0 -1020.0
510.0 255.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 2.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 255.0 765.0 1020.0 1020.0 1020.0 765.0 255.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 255.0 765.0 1020.0 1020.0 765.0 255.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 255.0 510.0 255.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 -255.0 -765.0 -1020.0 -1020.0 -765.0 -255.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 -255.0 -765.0 -1020.0 -1020.0 -1020.0 -1020.0 -765.0 -256.0 -2.0 -1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -1.0 -2.0 -1.0 0.0 0.0
0.0 2.0 4.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
510.0 763.0 1016.0 1018.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0 1020.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
#8) gradient magnitude
Sx = cv2.Sobel(gray,cv2.CV_64F,1,0,ksize=5)
Sy = cv2.Sobel(gray,cv2.CV_64F,0,1,ksize=5)
plt.subplot(1,2,1),plt.imshow(Sx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(1,2,2),plt.imshow(Sy,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
plt.show()
mag,angle=cv2.cartToPolar(Sx,Sy,angleInDegrees=True)
cv2.imshow("Magnitude",mag);
cv2.imshow("Direction",angle);
cv2.waitKey();
```

Figure 1



```
#9)gradient magnitude and direction in the matrix form
print("Gradient Magnitude Matrix: ")
gmag=np.array(mag)
for i in range(16):
    for j in range(16):
        print(gmag[i][j], end=" ")
    print()

print("Gradient Direction Matrix: ")
gang=np.array(angle)
for i in range(16):
    for j in range(16):
        print(gang[i][j], end=" ")
    print()
```

```
Gradient Magnitude Matrix:
0.0 1020.0 510.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
3570.0 5701.973342624464 7504.108874476782 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0
510.0 5701.973342624464 4342.493523311233 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4079.0001225798616 4076.0004986771046 4074.0 4076.0004986771046 4079.000122579
1530.0 7717.708434714993 3833.4985764850914 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.830951894845301 15.620499351813308 18.0 15.620499351813308 5.830951894845301 0.0
0.0 8160.0 4080.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 11.40175425099138 25.059928172283335 18.0 25.059928172283335 11.40175425099138 0.0
0.0 8160.0 4080.0 360.62445840513925 1486.8927331855514 2987.447334002802 3833.490576485092 4080.0 3833.490576485092 2987.447334002802 1480.093240306164 343.0838803172895
0.0 8160.0 4080.0 1140.3946685248927 4342.493523311233 7876.157057855055 10606.283514973566 11987.712459014023 11545.616051125206 9243.530169799847 5398.06502369136 1459
0.0 8160.0 4080.0 1803.1222920256962 6620.1850427310565 9974.379679960051 10606.283514973566 11774.264308227499 12003.974341858615 12308.868753870114 9960.88480005667 37
0.0 8160.0 4080.0 2040.0 7650.000000000001 10709.999999999998 6630.000000000001 1530.0 2600.4999519323205 8906.767090252219 11123.18461592722 5939.333632656188 1140.3946
0.0 8160.0 4080.0 1803.1222920256962 6620.1850427310565 9974.379679960051 10606.283514973566 11774.264308227499 11545.616051125206 9974.379679960051 9230.29121967449 588
0.0 8160.0 4080.0 1140.3946685248927 4342.493523311233 7876.157057855055 10606.283514973566 11987.712459014023 11987.712459014023 10606.283514973566 7873.445243348048 43
4.0 8156.003923498964 4080.0176470206598 352.1505359927768 1484.1489143613587 2987.447334002802 3833.490576485092 4080.0 4080.0 3833.490576485092 2989.0314539378915 1492
8.0 8144.015717077173 4080.07058762468 22.627416997969522 8.944271909999916 0.0 0.0 0.0 0.0 0.0 1.4142135623730951 4.472135954999958 6.0 4.472135954999958 1.4142135623730951
2550.0 8217.023548706671 5409.366876077089 4080.07058762468 4080.0176470206598 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0
5091.999999999999 8632.359584725371 8217.023548706671 8144.015717077173 8156.003923490964 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0
0.0 5092.0 2550.0 8.0 4.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Gradient Direction Matrix:
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
90.0 63.43289566040039 80.21784210205078 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.0
270.0 10.30475902557373 49.76069259643555 90.0 90.0 90.0 90.0 90.0 90.0 90.0 90.01404571533203 90.02810668945312 90.0 89.97189331054688 89.98595428466797 90.0
270.0 352.40606689453125 356.1866149902344 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 210.96006774902344 230.19302368164062 270.0 309.8069763183594 329.0399475097656 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 195.25729370117188 208.61809216308594 270.0 331.3898925781 344.7427062988281 0.0
0.0 0.0 0.0 224.99044799804688 239.03993225097656 254.74270629882812 266.1866149902344 270.0 273.8133850097656 285.2572937011719 300.4429626464844 318.9872741699219 270.
0.0 0.0 0.0 206.56710815429688 220.2393035888672 240.9463653564453 260.3114929199219 268.7813720703125 276.3393859863281 294.4476623535156 314.87268066640625 328.39450073
0.0 0.0 0.0 188.12948608398438 195.64462280273438 212.46551513671875 242.81748962402344 265.0310363769531 282.26544189453125 309.9595031738281 327.4851379394531 331.4668
0.0 0.0 0.0 180.0 180.0 180.0 180.0 180.0 348.6897888183594 346.75836181640625 344.0294189453125 339.9871350097656 333.4328918457031 0.0 0.0 0.0
0.0 0.0 0.0 171.87051391601562 164.35537719726562 147.53448486328125 117.18251037597656 94.96895599365234 83.66060638427734 57.534481048583984 24.491809844970703 7.50708
0.0 0.0 0.0 153.43289184570312 139.7606964111328 119.05363464355469 99.68850708007812 91.21862030029297 88.78137969970703 80.31149291992188 61.02682876586914 40.5494270
270.0 359.9438171386719 359.8315124511719 135.46197509765625 120.9336166381836 105.25729370117188 93.81339263916016 90.0 90.0 86.18660736083984 74.85331726074219 59.6054
270.0 359.887451171875 359.6630554199219 315.0095520019531 333.4328918457031 0.0 0.0 0.0 0.0 0.0 135.00955200195312 116.56710815429688 90.0 63.43289566040039 44.99045562
270.0 338.1319885253906 315.0095520019531 270.33697509765625 270.1684875488281 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0
270.0 315.0095520019531 291.8680114746094 270.112548828125 270.0561828613281 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0 270.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
gradient: max= 12308.868753870114 , min= 0.0
angle: max= 359.9438171386719 , min= 0.0
```

```
#10) non-maximal suppression of gradient magnitude
weak_th = None
strong_th = None
mag_max = np.max(mag)
print('gradient: max=',np.max(mag), ', min=',np.min(mag))
print('angle: max=',np.max(angle), ', min=',np.min(angle))
if not weak_th: weak_th = mag_max * 0.1
if not strong_th: strong_th = mag_max * 0.5

height, width = Img.shape
for i_x in range(width):
    for i_y in range(height):
        grad_ang = ang[i_y, i_x]
        grad_ang = abs(grad_ang - 180) if abs(grad_ang) > 180 else
abs(grad_ang)

        # In the x axis direction
```

```
        # gradient angle <= 45/2 means angle is approximated as 0,
        if grad_ang <= 22.5:
            neighb_1_x, neighb_1_y = i_x - 1, i_y
            neighb_2_x, neighb_2_y = i_x + 1, i_y

        # top right (diagnol-1) direction,
        # angle is approx 45, so edge is left diagonal like "\", so look top-
left and bot-right pixels
        elif grad_ang > 22.5 and grad_ang <= (22.5 + 45):
            neighb_1_x, neighb_1_y = i_x - 1, i_y - 1
            neighb_2_x, neighb_2_y = i_x + 1, i_y + 1

        # In y-axis direction
        # angle is approx 90, so edge is horizontal, so look right, left
        elif grad_ang > (22.5 + 45) and grad_ang <= (22.5 + 90):
            neighb_1_x, neighb_1_y = i_x, i_y - 1
            neighb_2_x, neighb_2_y = i_x, i_y + 1

        # top left (diagnol-2) direction
        # angle is approx 135, so edge is right diagonal like "/", so look
top-right and bot-left
        elif grad_ang > (22.5 + 90) and grad_ang <= (22.5 + 135):
            neighb_1_x, neighb_1_y = i_x - 1, i_y + 1
            neighb_2_x, neighb_2_y = i_x + 1, i_y - 1

        # Now it restarts the cycle
        # angle is approx 180, so edge is vertical, so look up and down
        elif grad_ang > (22.5 + 135) and grad_ang <= (22.5 + 180):
            neighb_1_x, neighb_1_y = i_x - 1, i_y
            neighb_2_x, neighb_2_y = i_x + 1, i_y

            # Non-maximum suppression step
        if width > neighb_1_x >= 0 and height > neighb_1_y >= 0:
            if mag[i_y, i_x] < mag[neighb_1_y, neighb_1_x]:
                mag[i_y, i_x] = 0  # suppress the pixel as it is non-maximum
as compared to its neighbors
                continue

        if width > neighb_2_x >= 0 and height > neighb_2_y >= 0:
            if mag[i_y, i_x] < mag[neighb_2_y, neighb_2_x]:
                mag[i_y, i_x] = 0
cv2.imshow('After suppression', mag)
cv2.waitKey()
```

```
# 11) non-max supp matrix
print("Non-maximal Suppression Matrix: ")
nonmax = np.array(mag)
for i in range(16):
    for j in range(16):
        print(nonmax[i][j], end=" ")
    print()
```

```
Non-maximal Suppression Matrix:
0.0 1020.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0
0.0 5701.973342624464 0.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 0.0 4076.0004906771046 0.0 0.0 0.0 4080.0
0.0 7717.700434714993 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 18.0 0.0 5.830951894845301 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 25.059928172283335 0.0 25.059928172283335 0.0 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 4080.0 0.0 2907.447334002802 0.0 343.08308031728984 0.0 30.265491900843113 0.0 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 11987.712459014023 0.0 9243.530169799847 0.0 1459.5458197672317 0.0 32.0 0.0 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 12308.868753870114 0.0 3736.885601674207 0.0 32.0 0.0 0.0
0.0 8160.0 0.0 0.0 0.0 10709.999999999998 0.0 0.0 0.0 0.0 11123.18461592722 0.0 1140.3946685248927 0.0 16.0 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 11774.264308227499 0.0 9974.379679960051 0.0 5885.455462409005 0.0 30.265491900843113 0.0 0.0
0.0 8160.0 0.0 0.0 0.0 0.0 0.0 11987.712459014023 11987.712459014023 0.0 7873.445243348048 0.0 1148.5573559985488 0.0 11.40175425099138 0.0
0.0 8156.003923490964 0.0 0.0 0.0 0.0 0.0 4080.0 4080.0 0.0 2909.0314539378915 0.0 373.56927068483566 0.0 5.830951894845301 0.0
0.0 8144.015717077173 0.0 22.627416997969522 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6.0 0.0 1.4142135623730951 0.0
0.0 8217.023548706671 0.0 4080.07058762468 0.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0 4080.0
0.0 8632.359584725371 0.0 0.0 0.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0 8160.0
0.0 5092.0 0.0 8.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
#12) hysterisis threshold
# double thresholding step
for i_x in range(width):
    for i_y in range(height):

        grad_mag = mag[i_y, i_x]

        if grad_mag < weak_th:
            mag[i_y, i_x] = 0
        elif strong_th > grad_mag >= weak_th:
            mag[i_y, i_x] = 2  # weak
        else:
            mag[i_y, i_x] = 1  # strong
M, N = Img.shape
for i in range(1, M - 1):
    for j in range(1, N - 1):
        if (mag[i, j] == 2):
            if ((mag[i + 1, j - 1] == 1) or (mag[i + 1, j] == 1) or (mag[i +
1, j + 1] == 1)
                    or (mag[i, j - 1] == 1) or (mag[i, j + 1] == 1)
                    or (mag[i - 1, j - 1] == 1) or (mag[i - 1, j] == 1) or
(mag[i - 1, j + 1] == 1)):
                mag[i, j] = 1
            else:
                Img[i, j] = 0

cv2.imshow('After hysteresis', mag)
cv2.waitKey()
```

```python
# 13) Hysterisis matrix
print("Hysterisis Matrix: ")
hyst = np.array(mag)
for i in range(16):
    for j in range(16):
        print(hyst[i][j], end=" ")
    print()
```

```
Hysterisis Matrix:
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 2.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 2.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 2.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 2.0
0.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Process finished with exit code 0
```