영상 처리 관련 면접 문제

Vision and Display Systems Lab 지원자 윤준영

Contents

01. What is Edge Detection?

02. Backgrounds

03. Derivative Filters

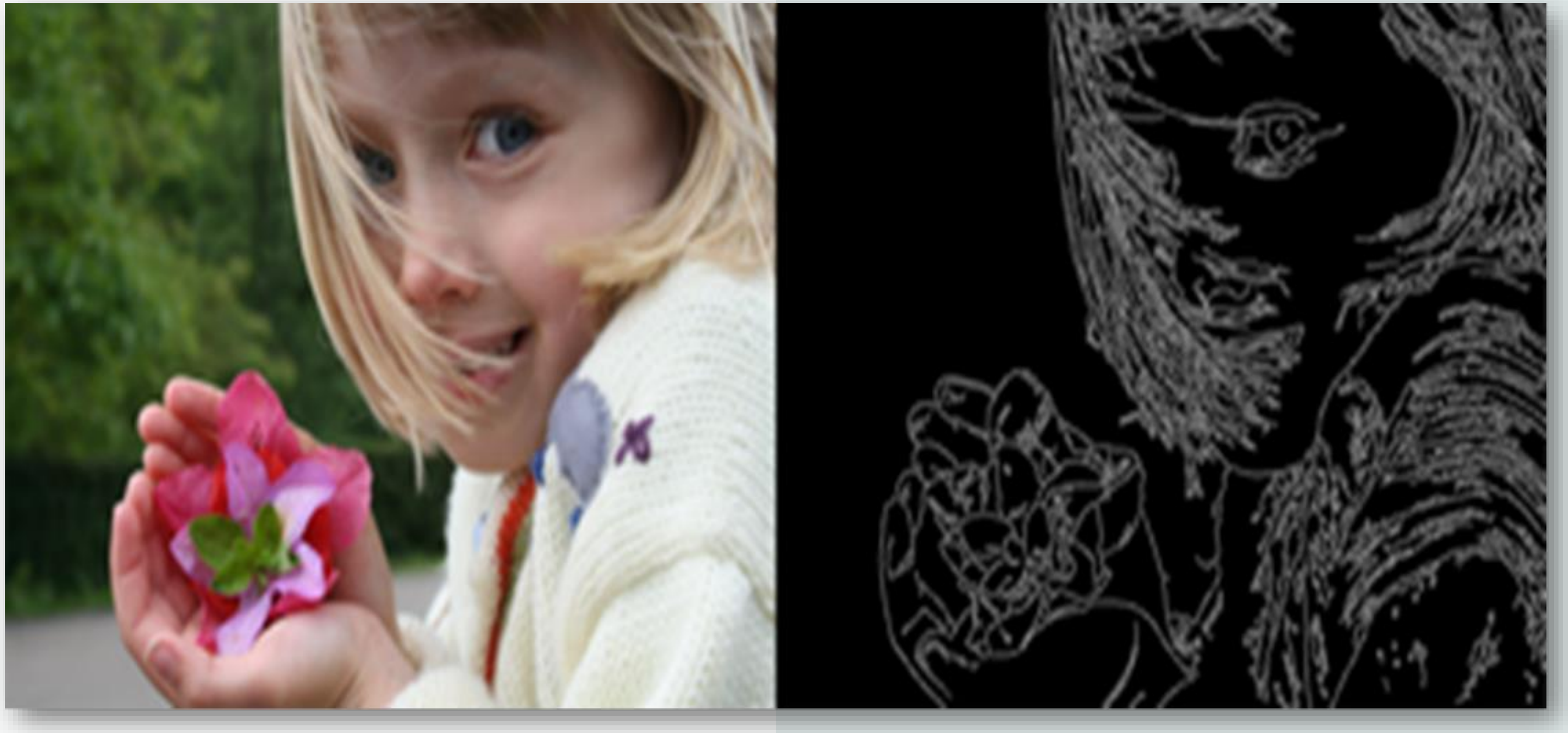
04. Non Maximum Suppression

05. Hysteresis Thresholding

06. Experiment

07. What I tried

01. What is Edge Detection?



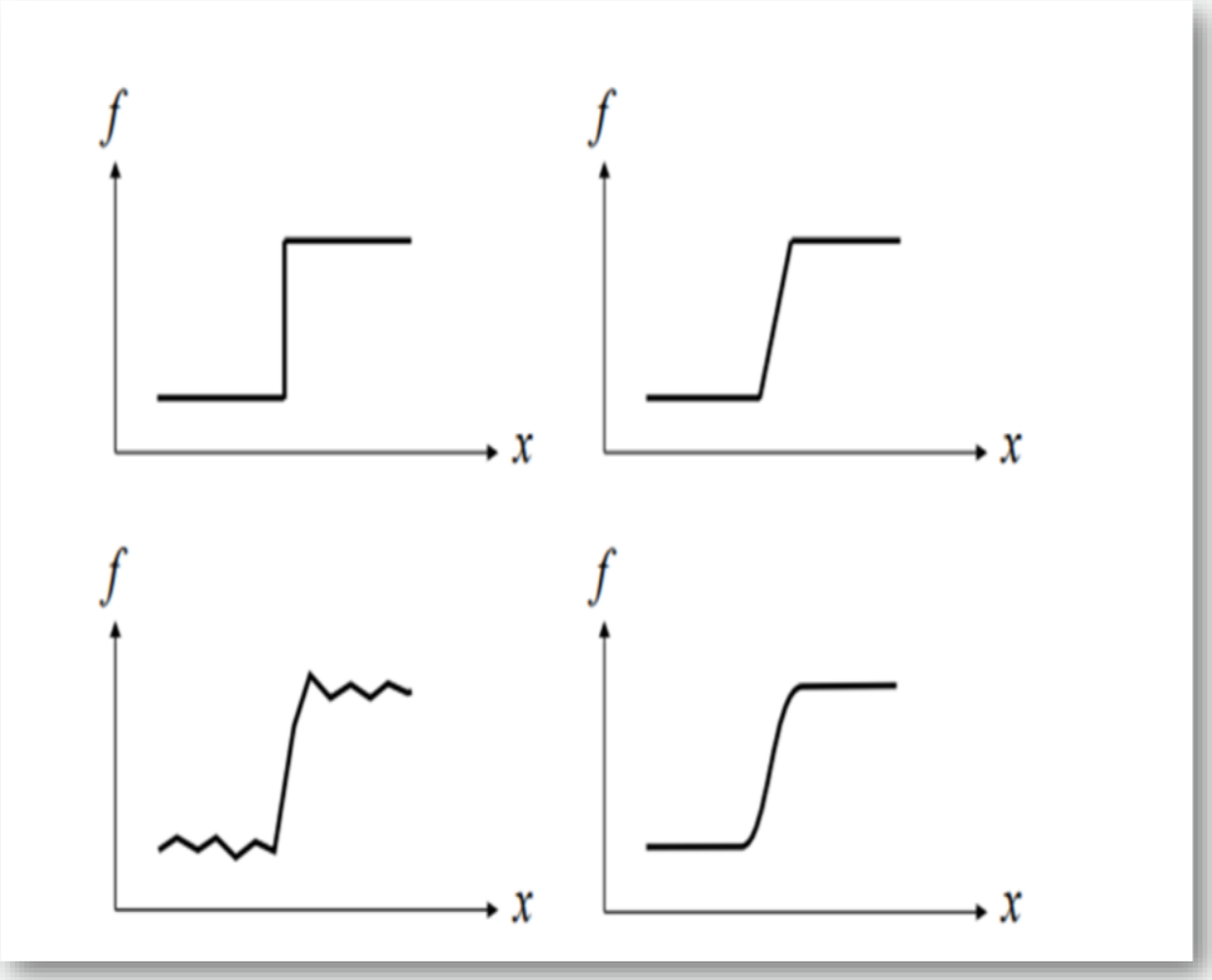
02. Backgrounds

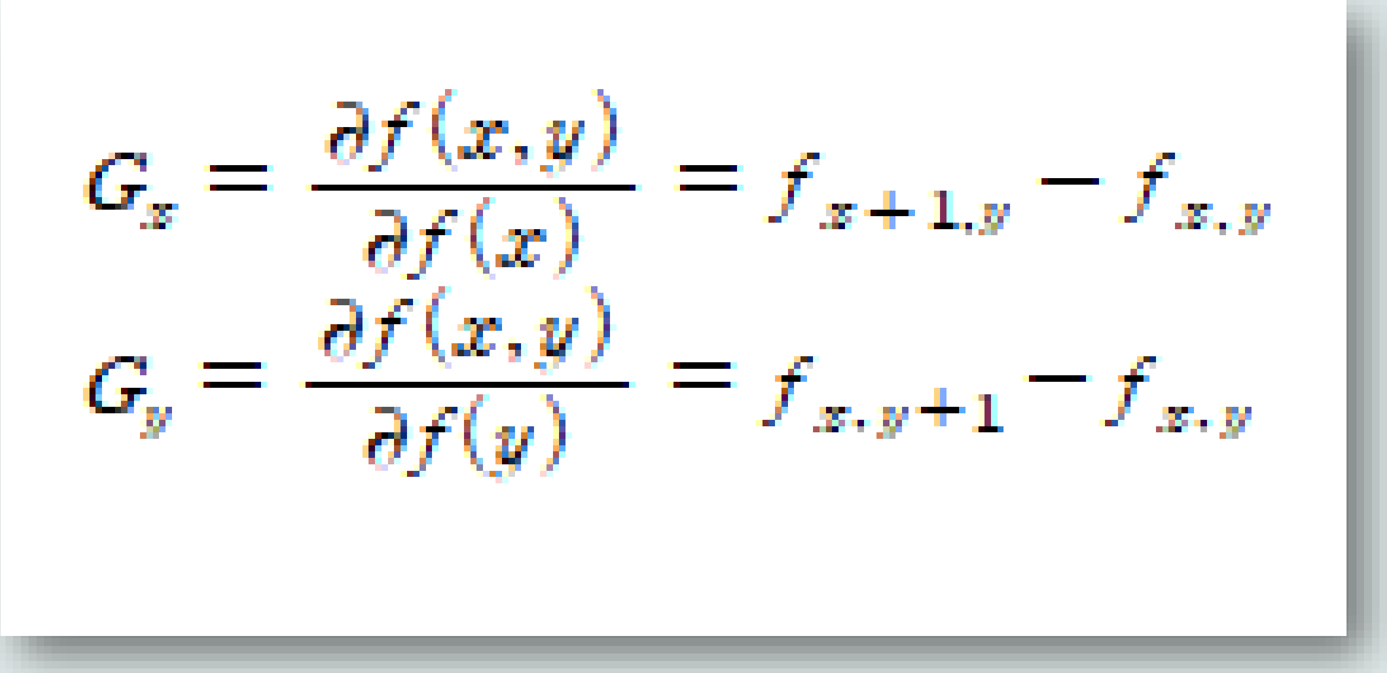
❶ Derivative in image processing

❷ Convolution in image processing

❸ Zero padding

1) Derivative in image processing





$$G_x = \frac{\partial f(x,y)}{\partial f(x)} = f_{x+1,y} - f_{x,y}$$

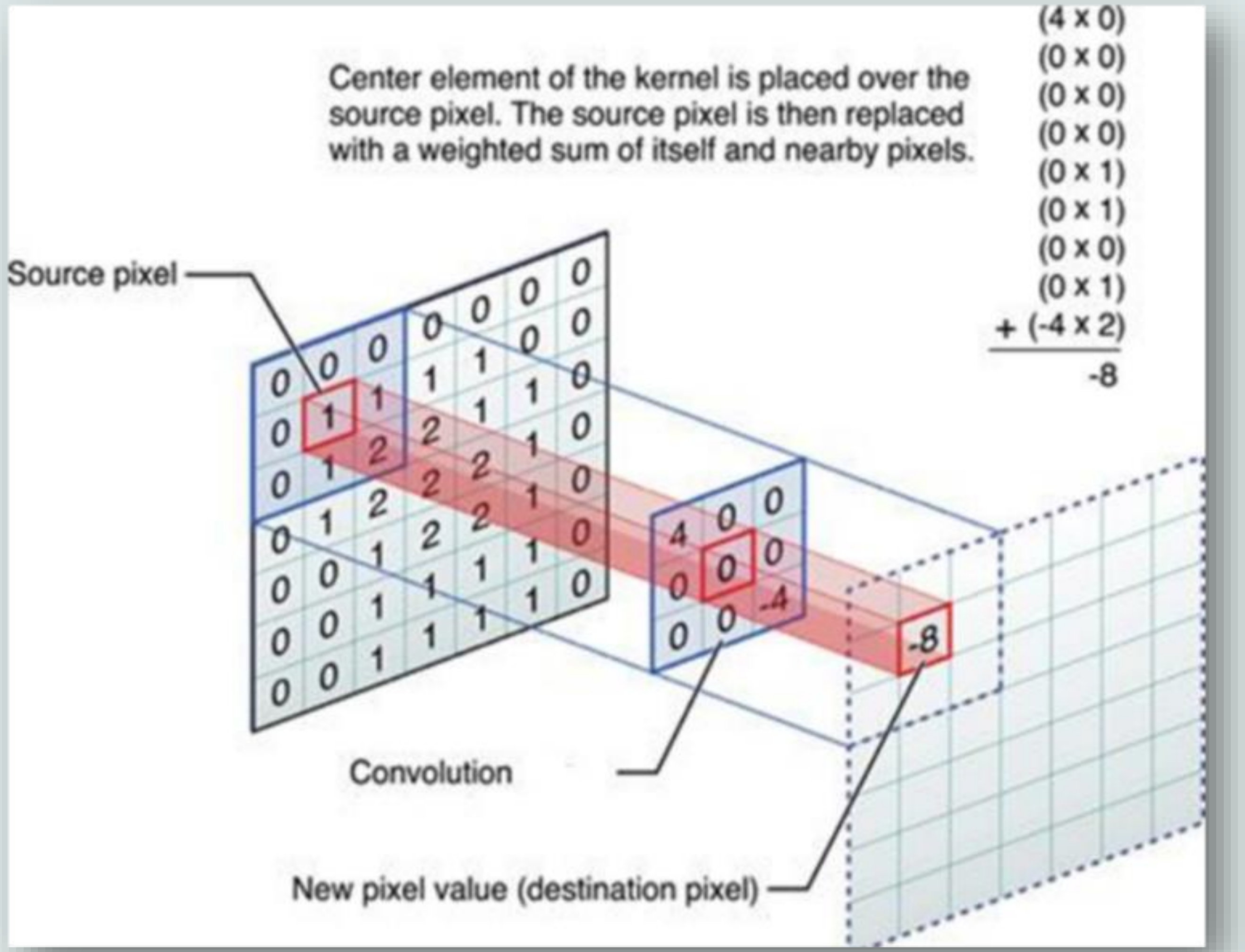
$$G_y = \frac{\partial f(x,y)}{\partial f(y)} = f_{x,y+1} - f_{x,y}$$

2) Convolution in image processing

☑ In mathematics...

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau$$

☑ In image processing...



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

3) Zero padding

4	0	0
0	0	0
0	0	-4

<Kernel>

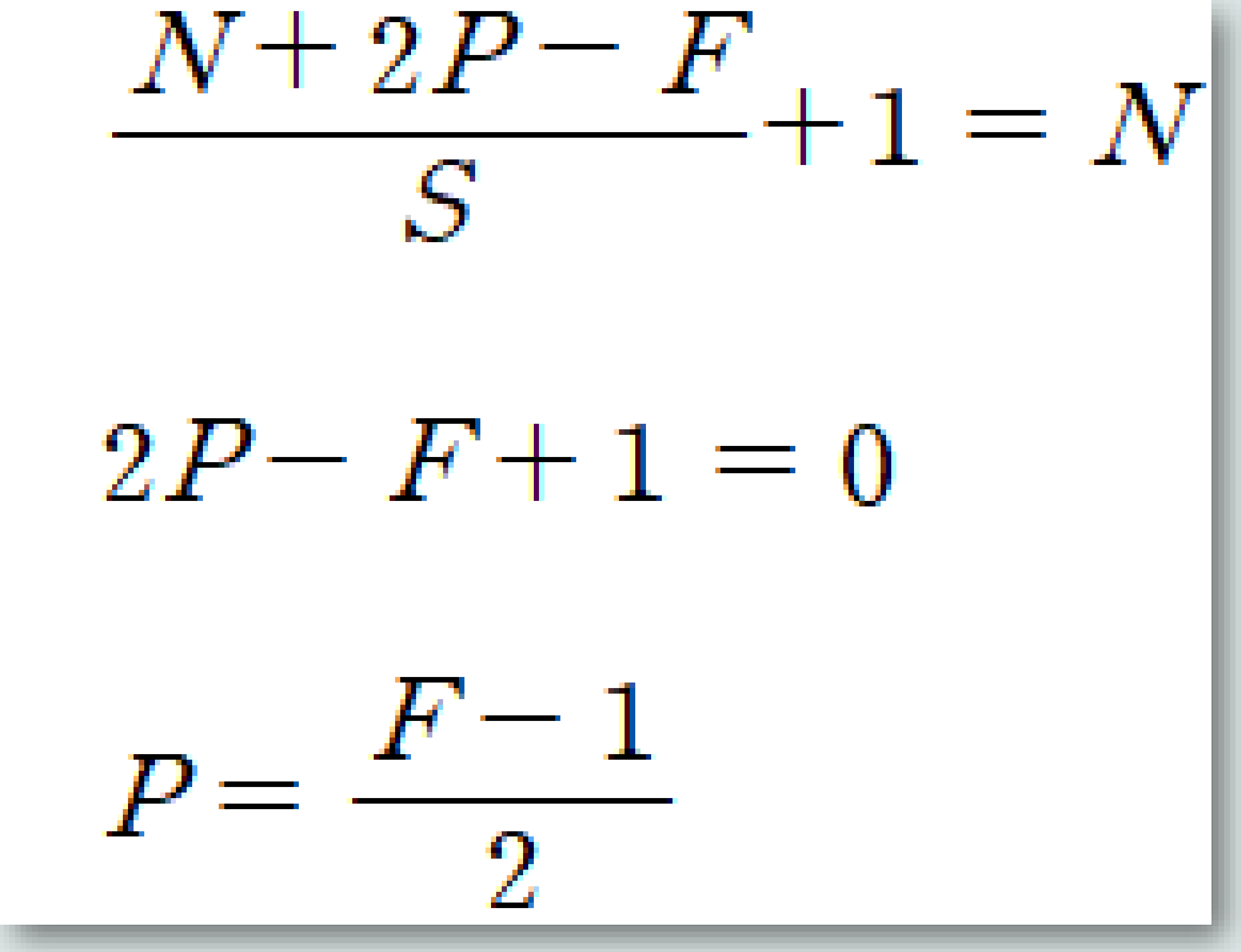
0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	1	2	1	1	1	0
0	1	2	2	2	1	0
0	0	1	1	1	1	0
0	0	1	1	1	1	0
0	0	1	1	1	1	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	2	2	1	1	0	0
0	0	1	2	2	2	1	0	0
0	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

-8	-8	-4	-4	0
-8	-4	-4	0	4
-4	-4	0	4	8
-4	0	4	4	8
-4	-4	0	4	8

-4	-4	-4	-4	0	0	0
-4	-8	-8	-4	-4	0	0
-4	-8	-4	-4	0	4	0
0	-4	-4	0	4	8	4
0	-4	0	4	4	8	4
0	-4	0	4	4	8	4
0	-4	-4	0	4	8	4

Padding formula



$$\frac{N+2P-F}{S}+1=N$$

$$2P-F+1=0$$

$$P=\frac{F-1}{2}$$

N : Shape of input image

P : Padding

F : Shape of filter

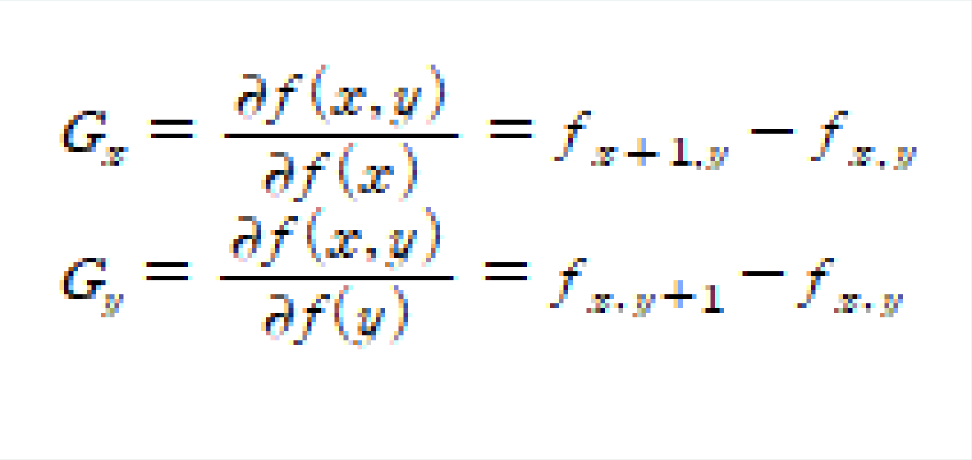
S : Stride

03. Derivative Filter

❶ Basic Derivative Filter

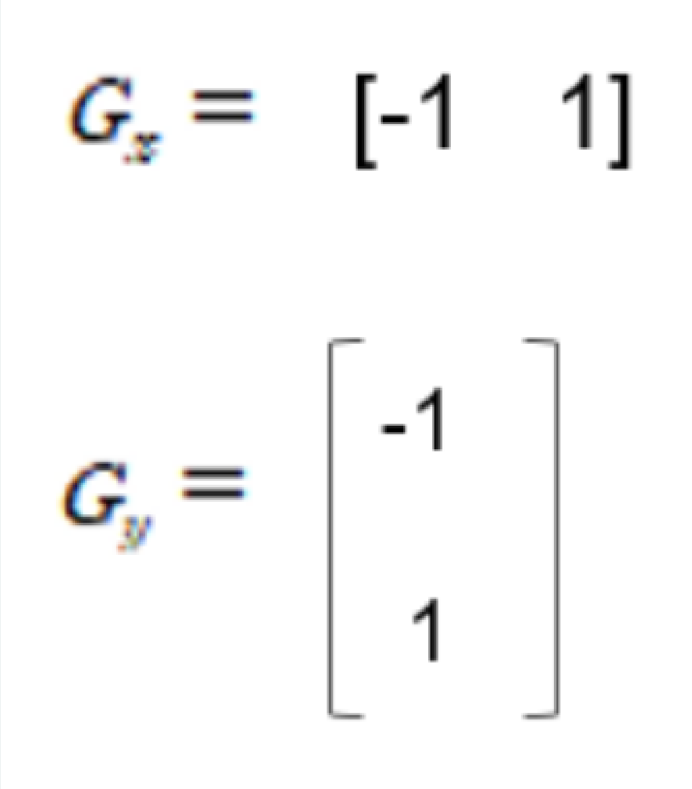
❷ Kinds of Derivative Filters

1) Basic Derivative Filter



$$G_x = \frac{\partial f(x,y)}{\partial f(x)} = f_{x+1,y} - f_{x,y}$$
$$G_y = \frac{\partial f(x,y)}{\partial f(y)} = f_{x,y+1} - f_{x,y}$$

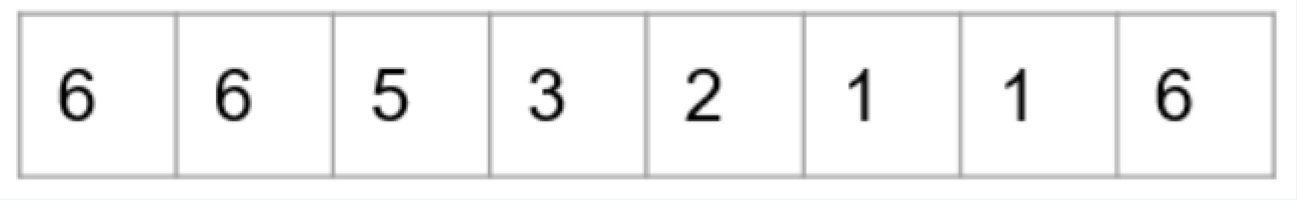
미분 연산 convolution kernel



$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

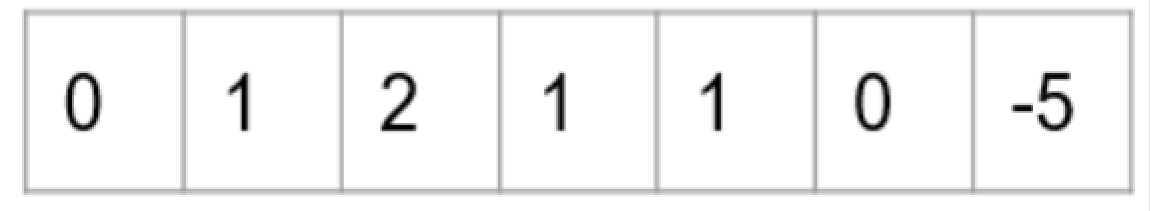
$$G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

현재 픽셀

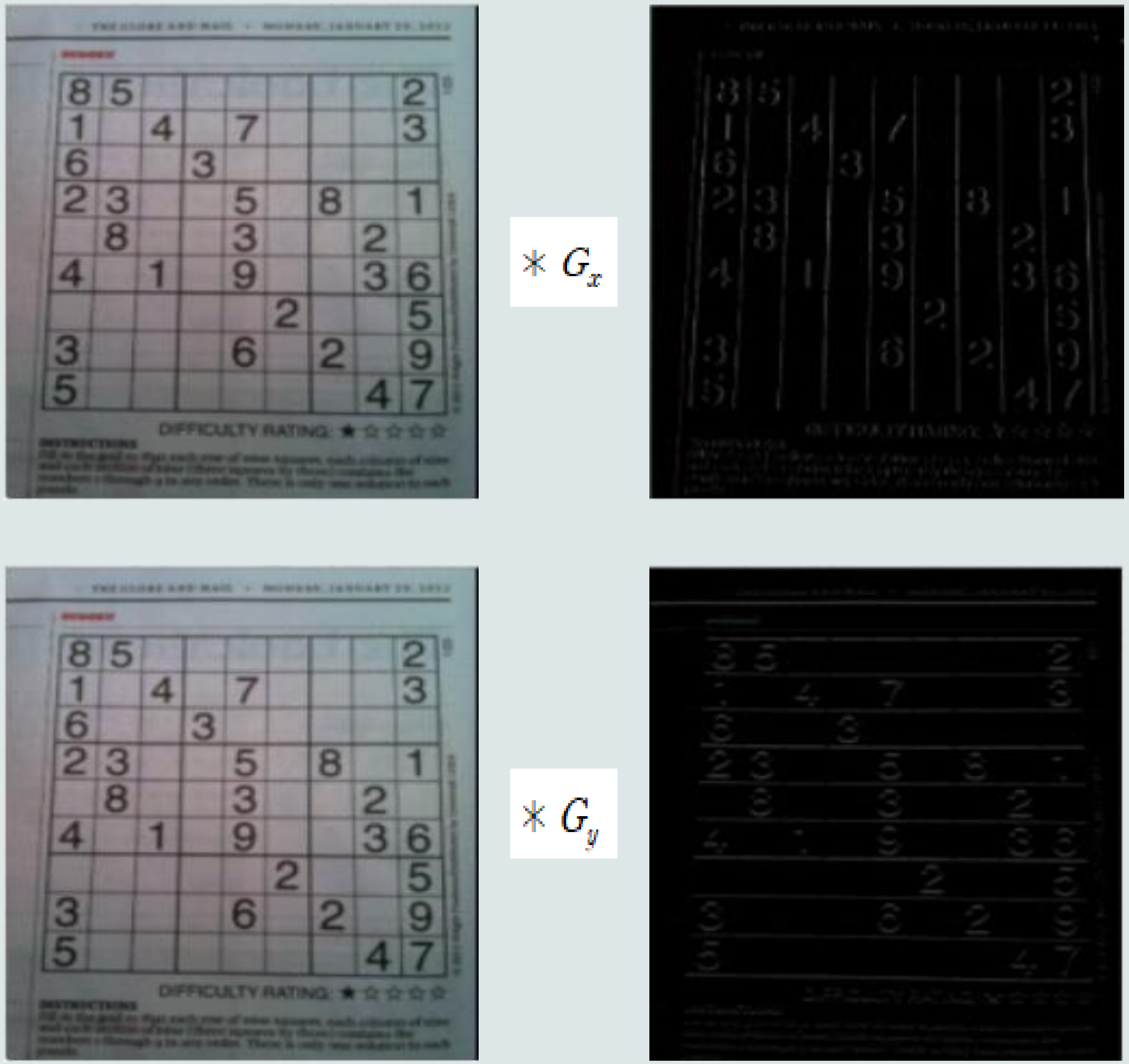


6	6	5	3	2	1	1	6

1차 미분



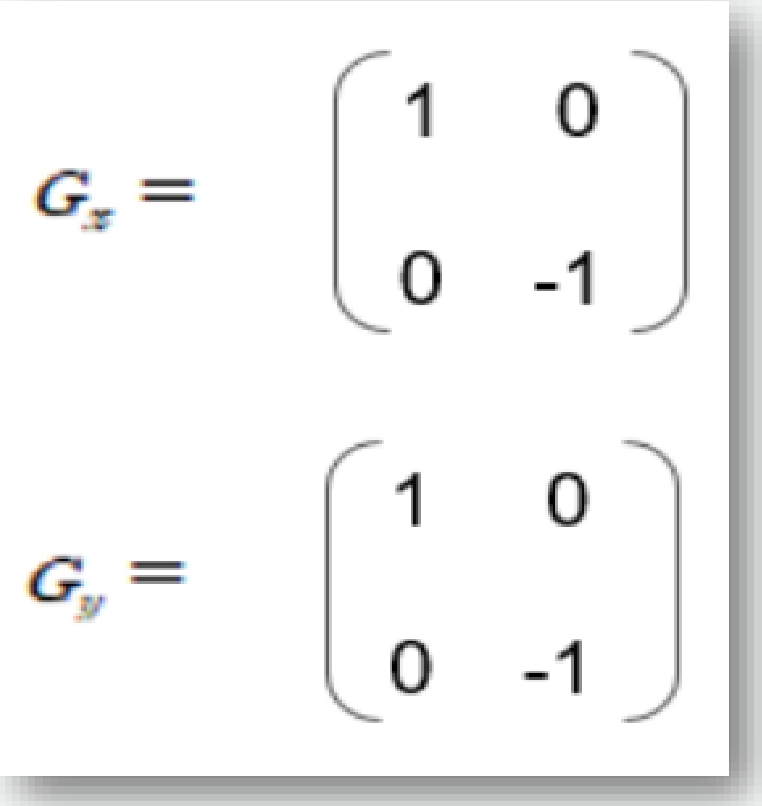
0	1	2	1	1	0	-5



$$magnitudc = \sqrt{G_x^2 + G_y^2}$$

$$dirction(\theta) = \arctan(\frac{G_y}{G_x})$$

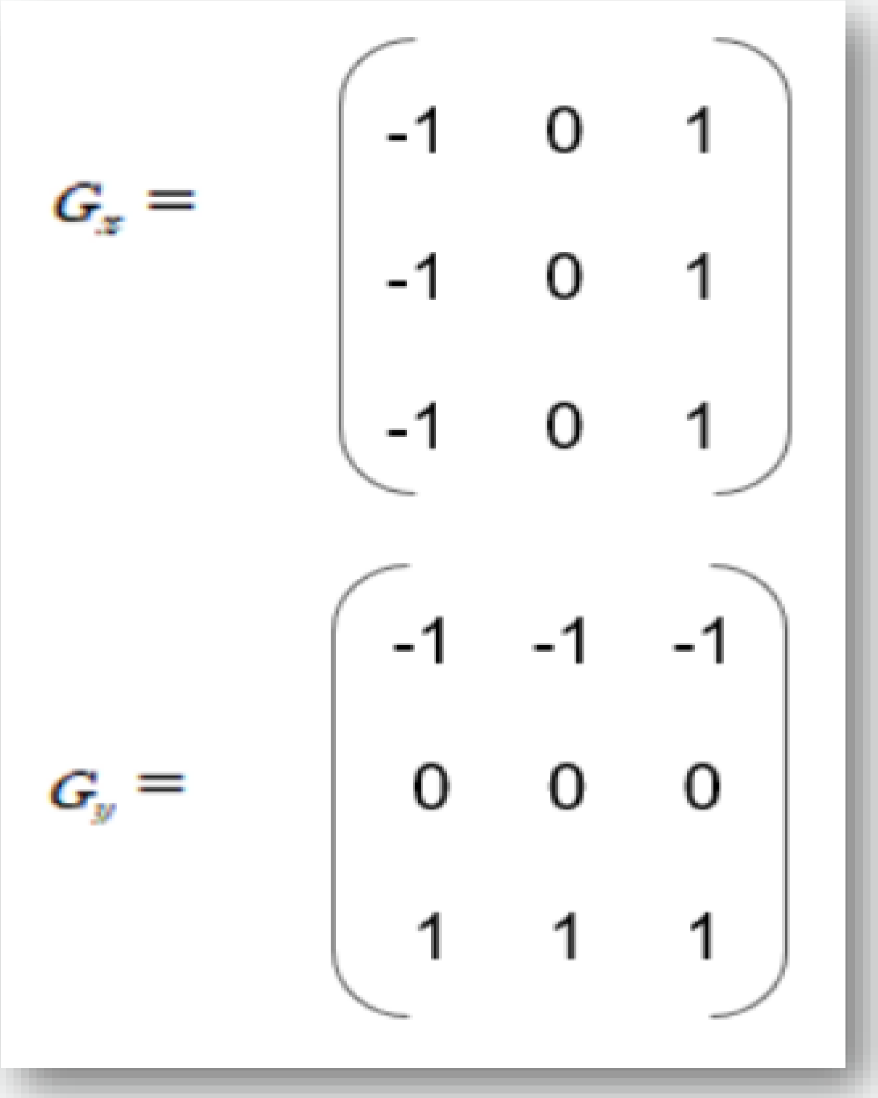
2) Kinds of Derivative Filters



$$G_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$G_y = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

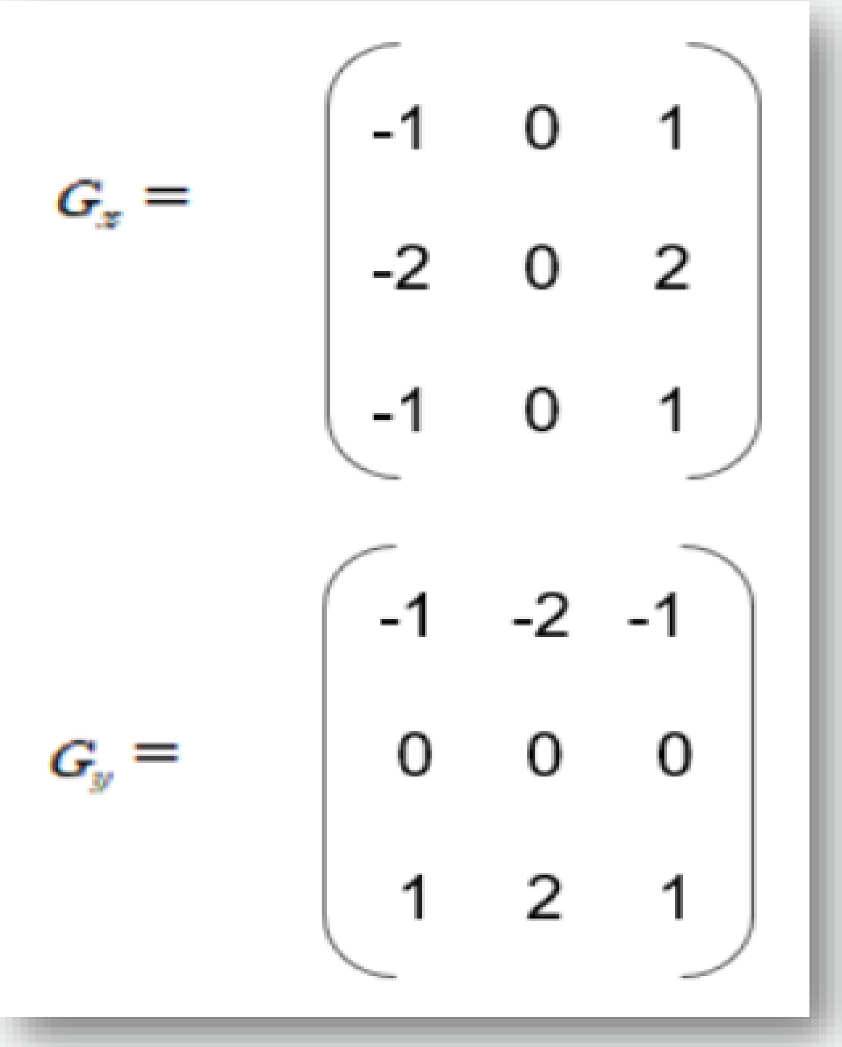
로버츠 교차 필터



$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

$$G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

프리윗 필터

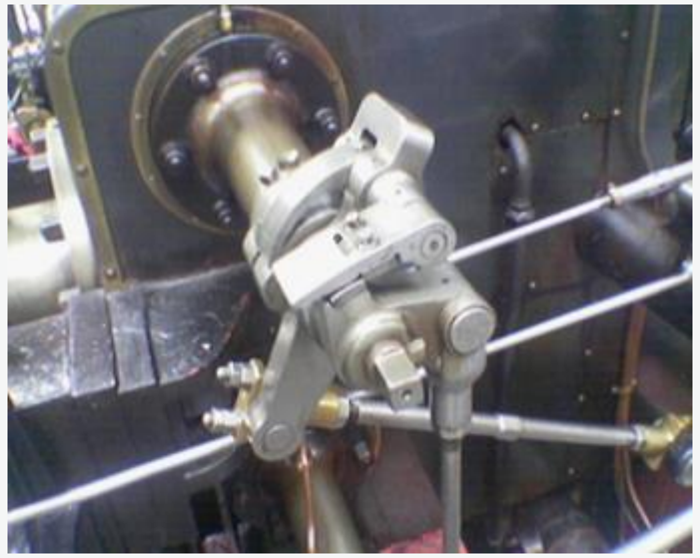


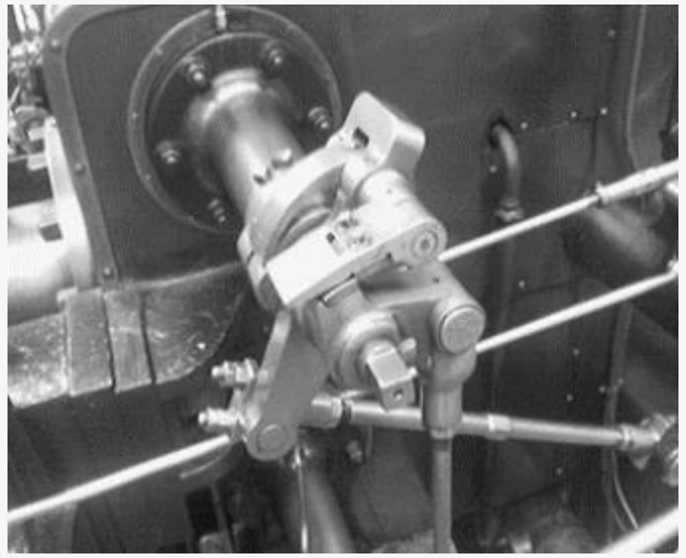
$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

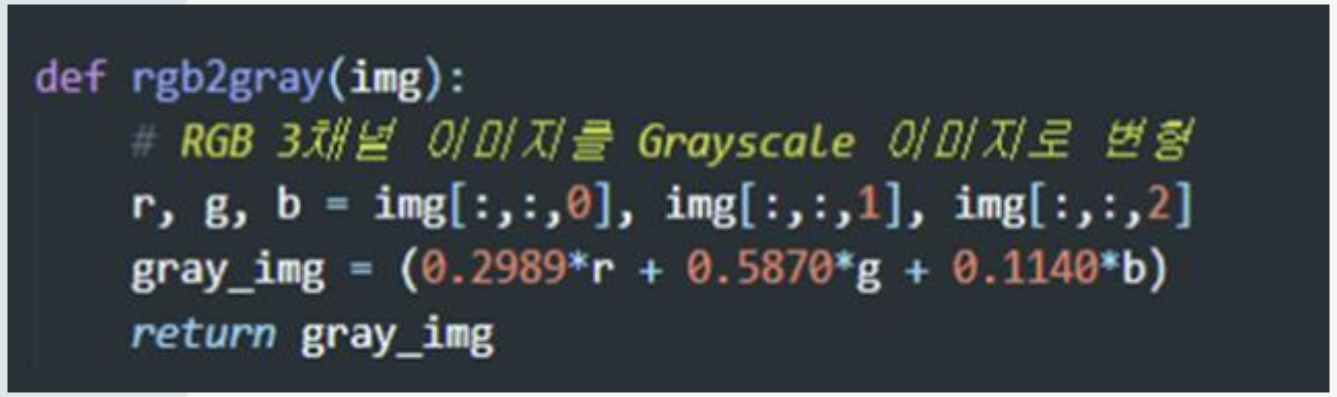
$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

소벨 필터

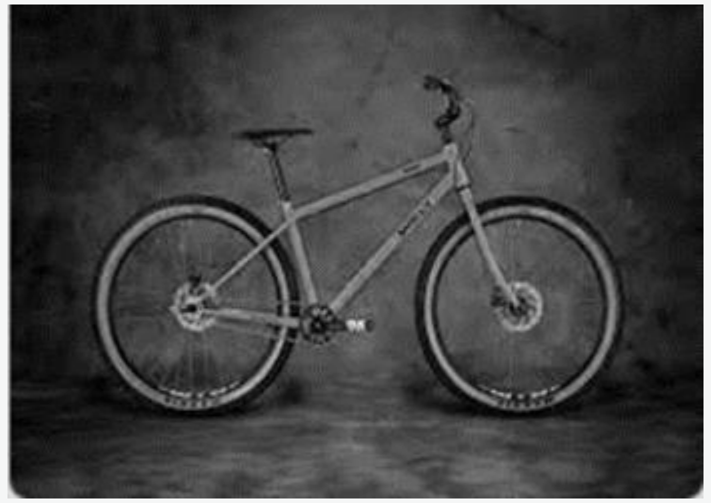
First Step: Convert RGB image to Grayscale image







```
def rgb2gray(img):
    # RGB 3채널 이미지를 Grayscale 이미지로 변형
    r, g, b = img[:,:,0], img[:,:,1], img[:,:,2]
    gray_img = (0.2989*r + 0.5870*g + 0.1140*b)
    return gray_img
```

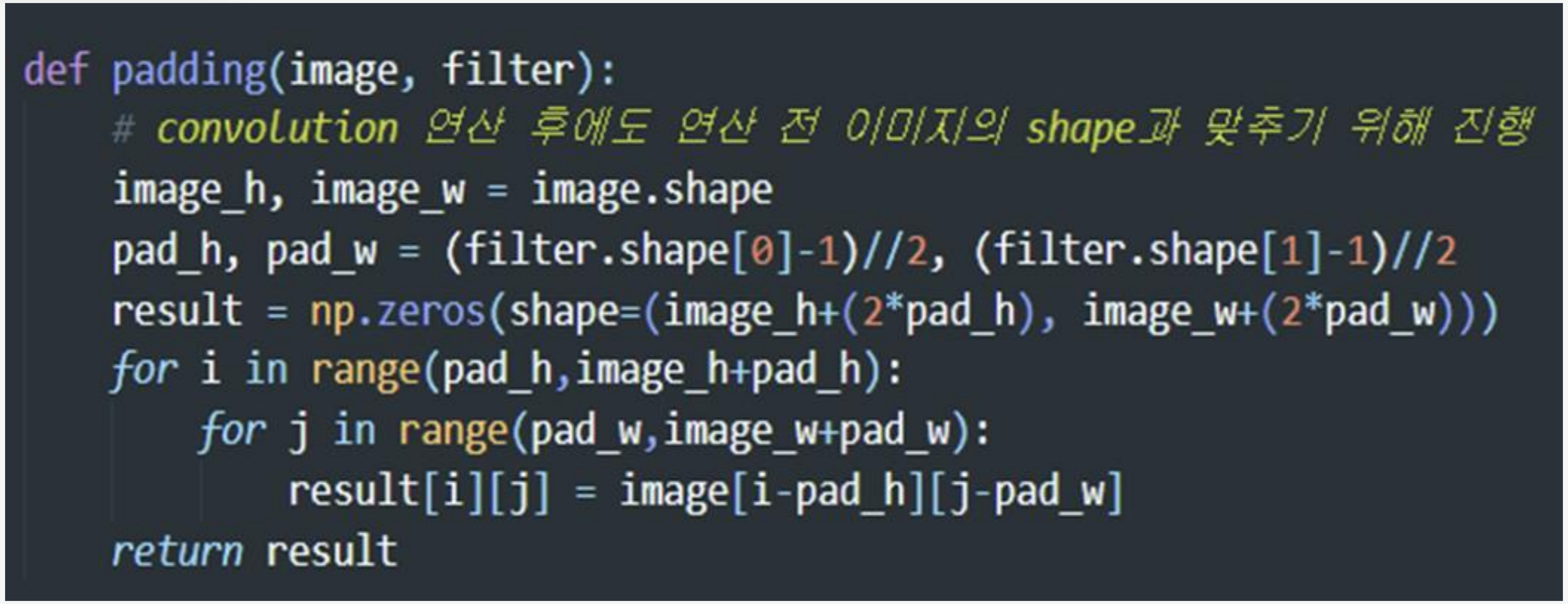


Second Step: Zero padding

$$\frac{N+2P-F}{S}+1=N$$

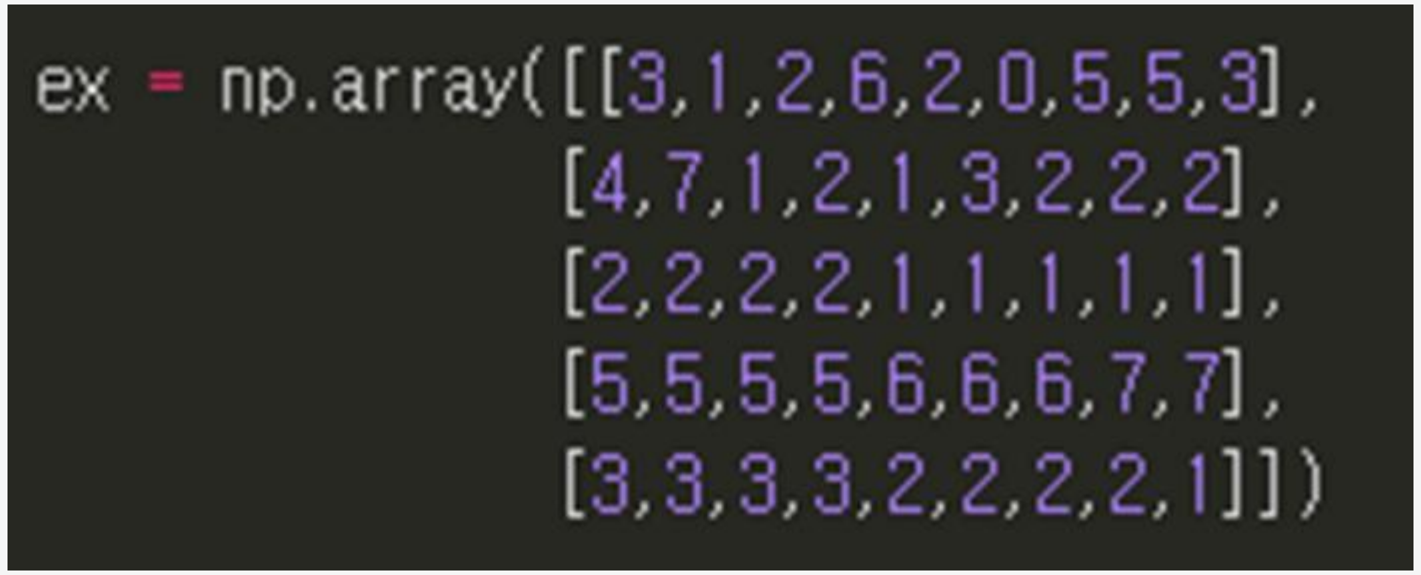
$$2P-F+1=0$$

$$P=\frac{F-1}{2}$$



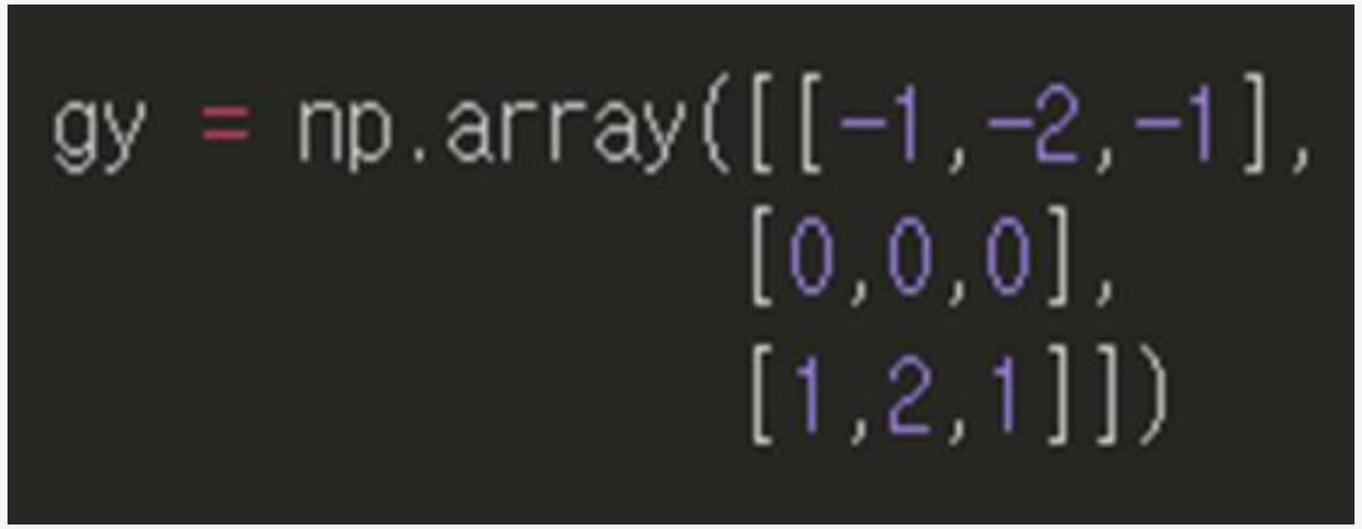
```python
def padding(image, filter):
    # convolution 연산 후에도 연산 전 이미지의 shape과 맞추기 위해 진행
    image_h, image_w = image.shape
    pad_h, pad_w = (filter.shape[0]-1)//2, (filter.shape[1]-1)//2
    result = np.zeros(shape=(image_h+(2*pad_h), image_w+(2*pad_w)))
    for i in range(pad_h,image_h+pad_h):
        for j in range(pad_w,image_w+pad_w):
            result[i][j] = image[i-pad_h][j-pad_w]
    return result
```

Image array

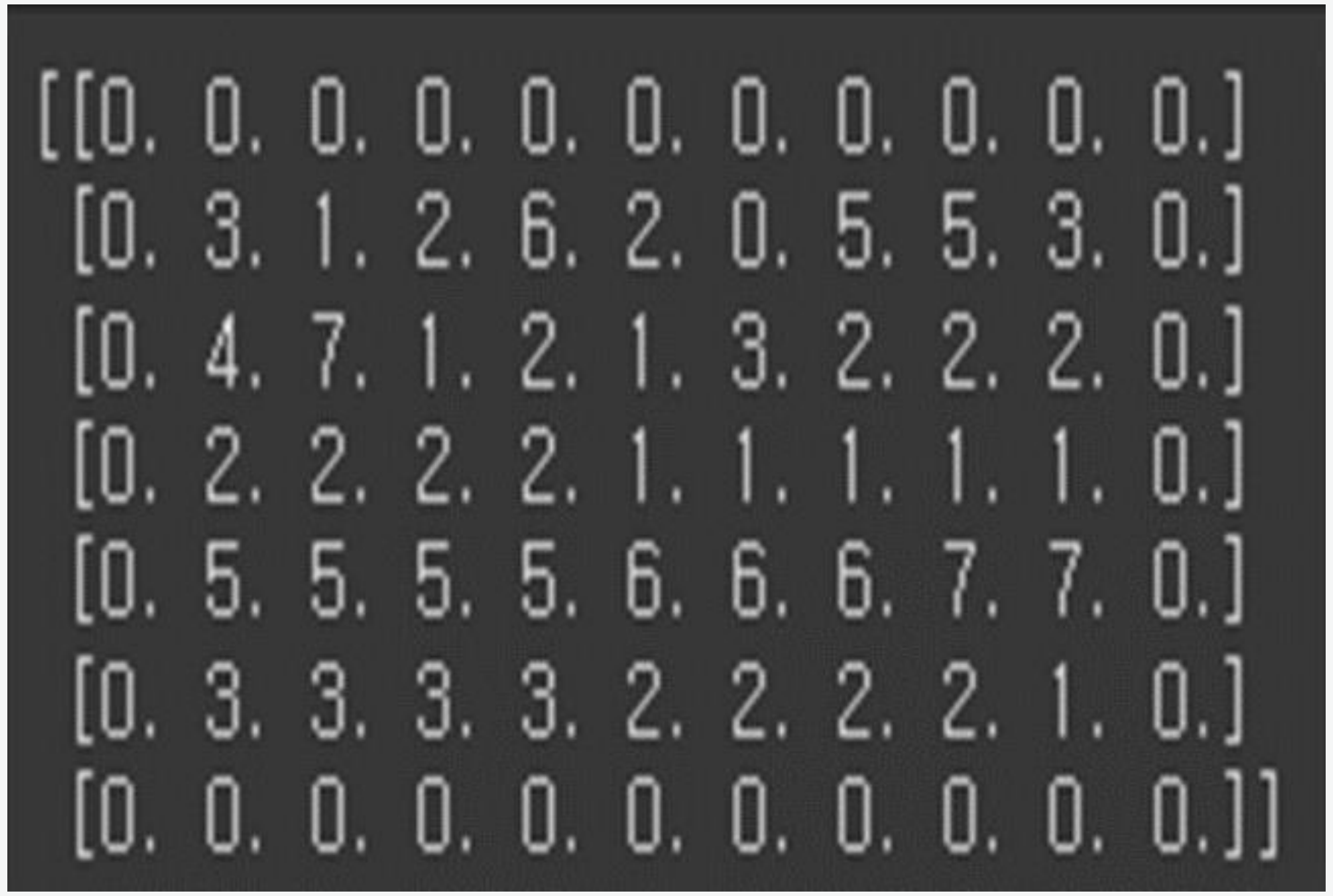


```
ex = np.array([[3,1,2,6,2,0,5,5,3],
               [4,7,1,2,1,3,2,2,2],
               [2,2,2,2,1,1,1,1,1],
               [5,5,5,5,6,6,6,7,7],
               [3,3,3,3,2,2,2,2,1]])
```

Filter array

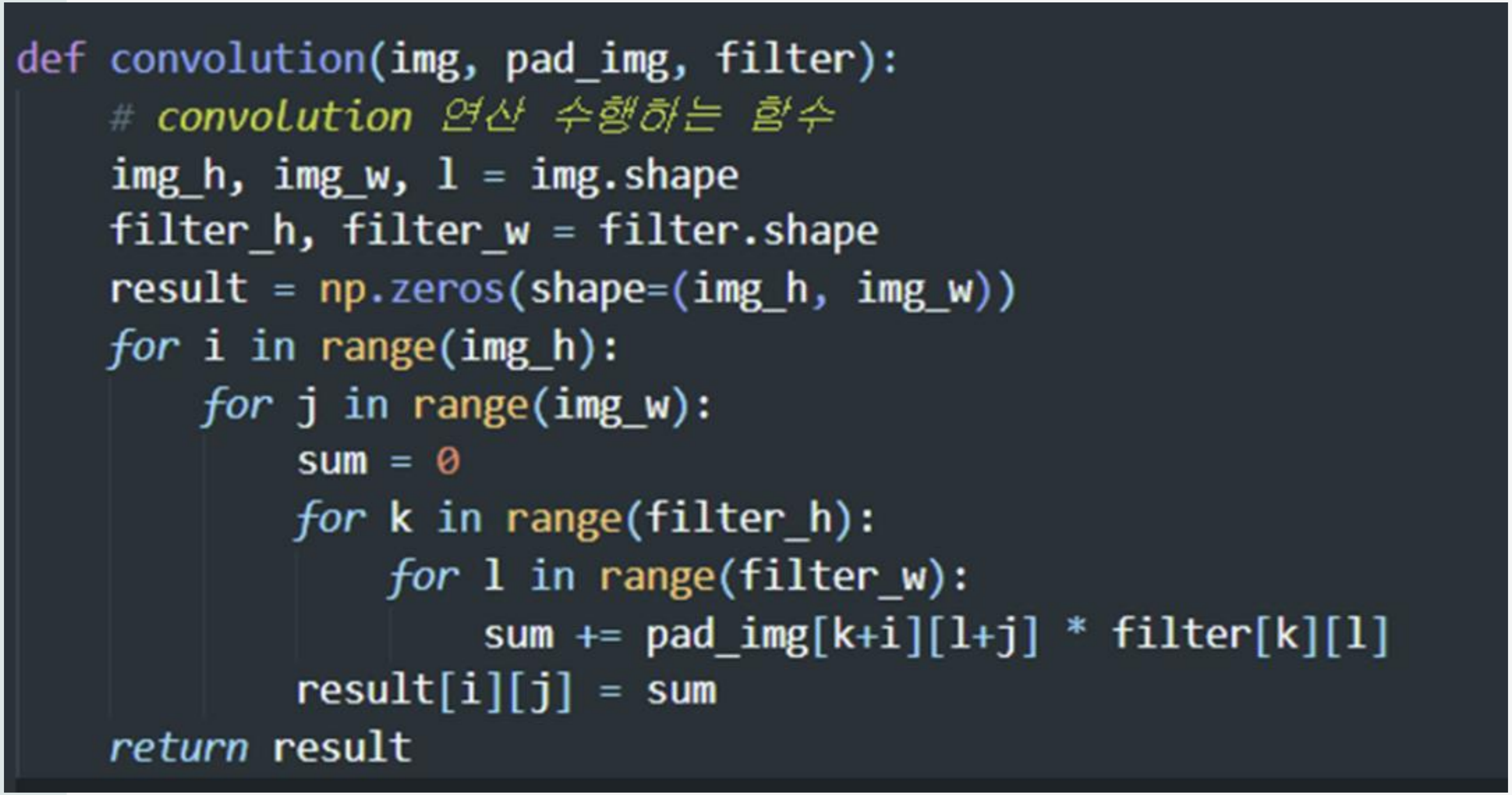


```
gy = np.array([[-1,-2,-1],
               [0,0,0],
               [1,2,1]])
```



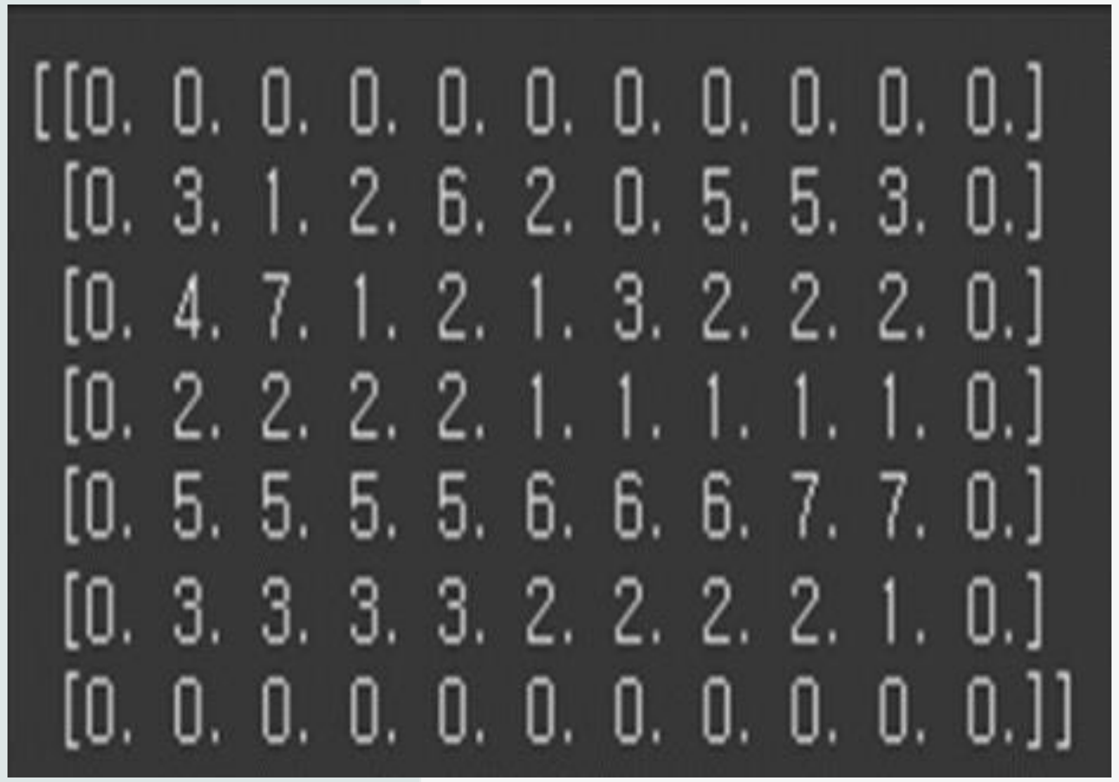
```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 3. 1. 2. 6. 2. 0. 5. 5. 3. 0.]
 [0. 4. 7. 1. 2. 1. 3. 2. 2. 2. 0.]
 [0. 2. 2. 2. 2. 1. 1. 1. 1. 1. 0.]
 [0. 5. 5. 5. 5. 6. 6. 6. 7. 7. 0.]
 [0. 3. 3. 3. 3. 2. 2. 2. 2. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Third Step: Convolution



```python
def convolution(img, pad_img, filter):
    # convolution 연산 수행하는 함수
    img_h, img_w, l = img.shape
    filter_h, filter_w = filter.shape
    result = np.zeros(shape=(img_h, img_w))
    for i in range(img_h):
        for j in range(img_w):
            sum = 0
            for k in range(filter_h):
                for l in range(filter_w):
                    sum += pad_img[k+i][l+j] * filter[k][l]
            result[i][j] = sum
    return result
```

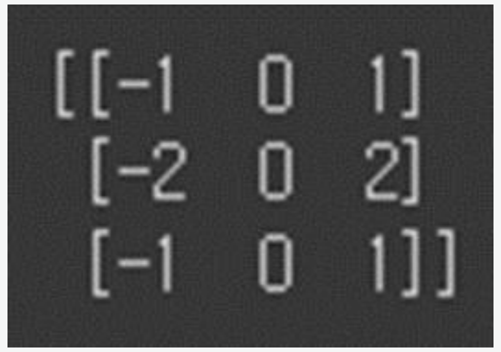
Pad array



```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 3. 1. 2. 6. 2. 0. 5. 5. 3. 0.]
 [0. 4. 7. 1. 2. 1. 3. 2. 2. 2. 0.]
 [0. 2. 2. 2. 2. 1. 1. 1. 1. 1. 0.]
 [0. 5. 5. 5. 5. 6. 6. 6. 7. 7. 0.]
 [0. 3. 3. 3. 3. 2. 2. 2. 2. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

*

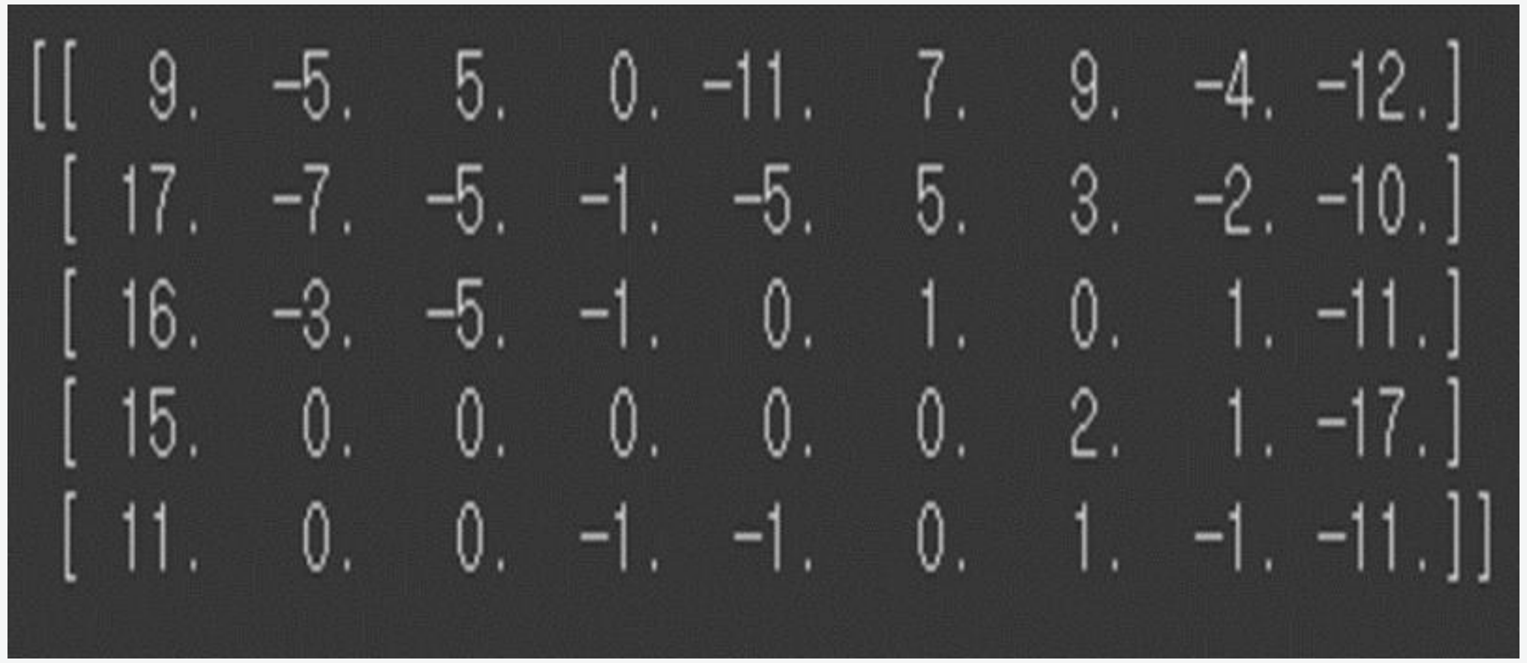
Kernel



```
[[-1  0  1]
 [-2  0  2]
 [-1  0  1]]
```

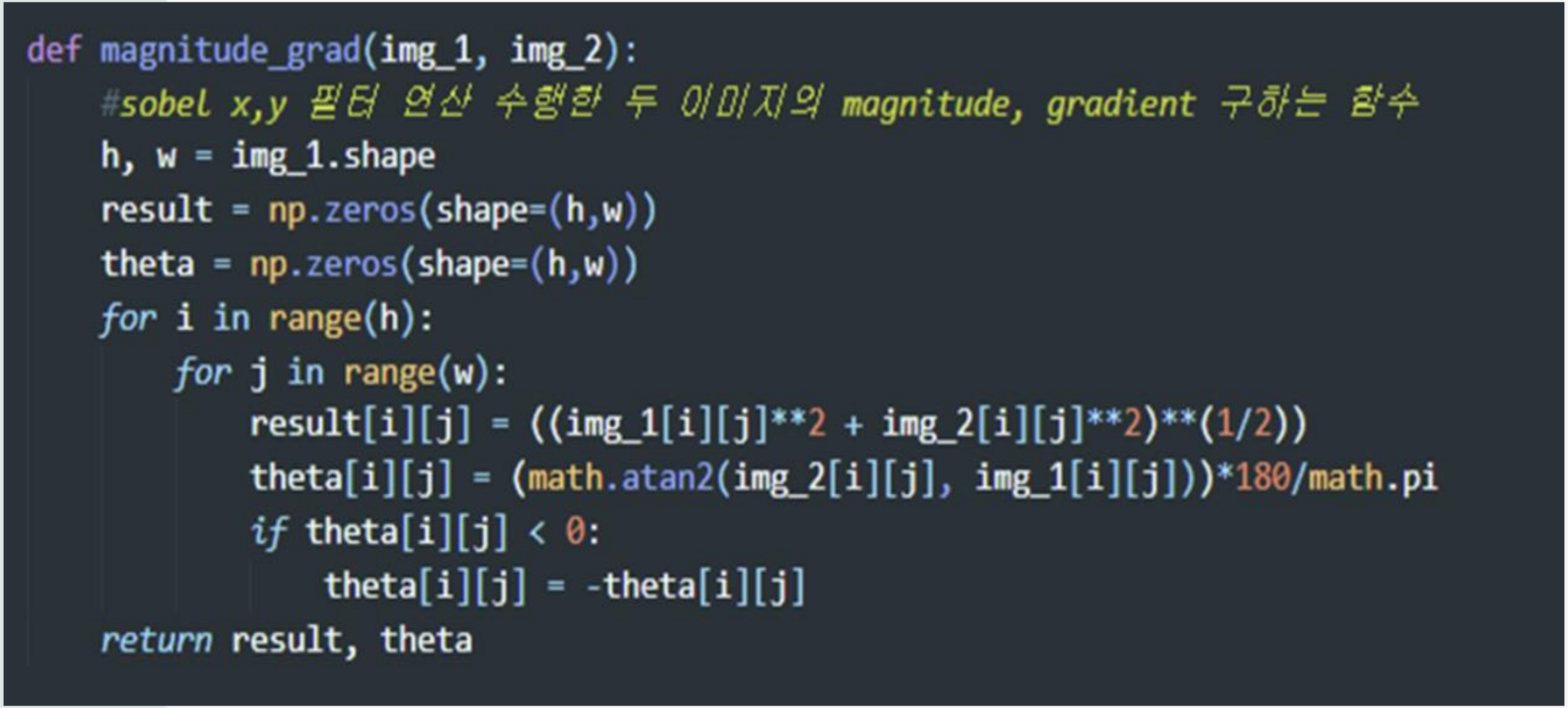
=

Result



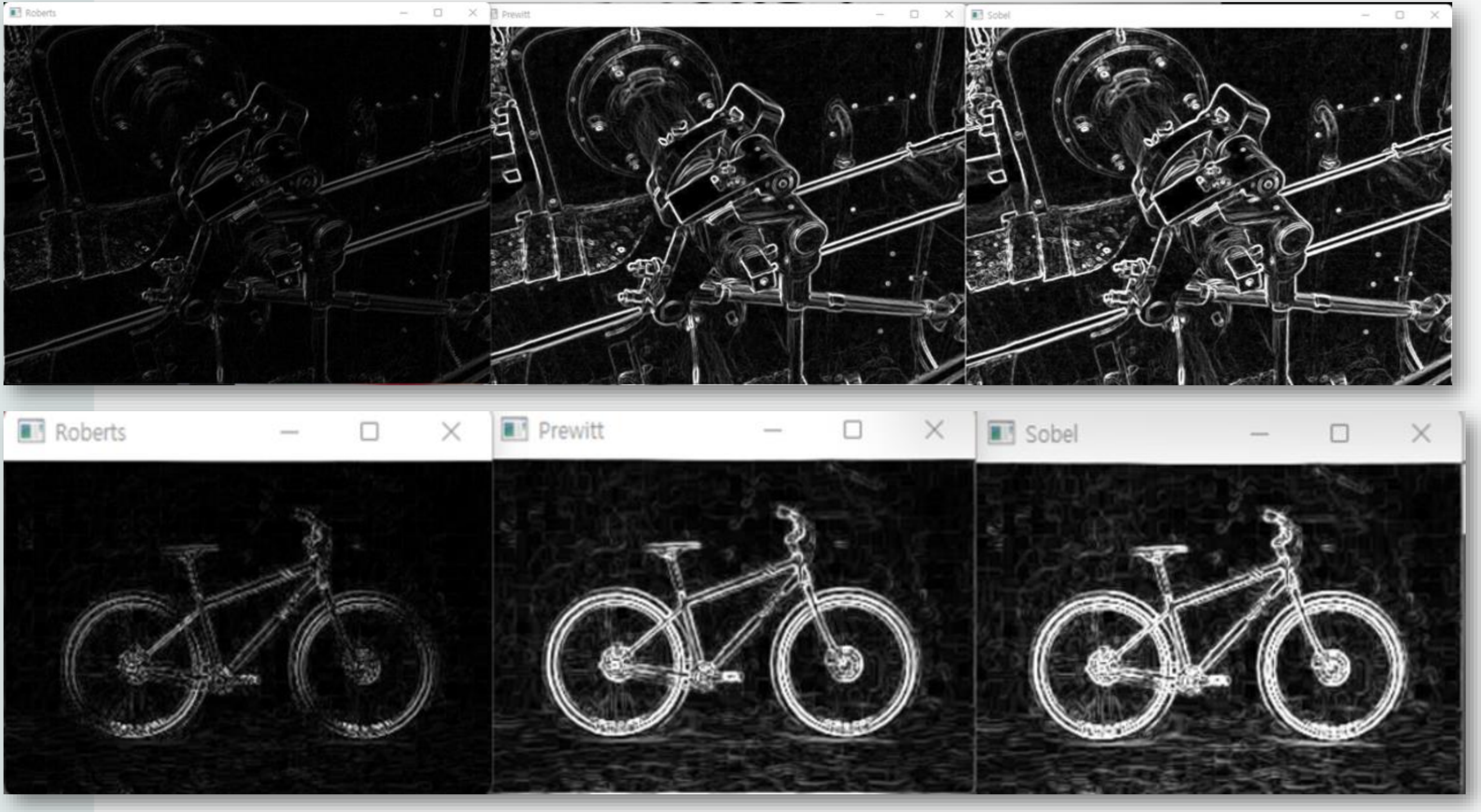
```
[[  9.  -5.   5.   0. -11.   7.   9.  -4. -12.]
 [ 17.  -7.  -5.  -1.  -5.   5.   3.  -2. -10.]
 [ 16.  -3.  -5.  -1.   0.   1.   0.   1. -11.]
 [ 15.   0.   0.   0.   0.   0.   2.   1. -17.]
 [ 11.   0.   0.  -1.  -1.   0.   1.  -1. -11.]]
```

Fourth Step: Magnitude and Gradient



```python
def magnitude_grad(img_1, img_2):
    #sobel x,y 필터 연산 수행한 두 이미지의 magnitude, gradient 구하는 함수
    h, w = img_1.shape
    result = np.zeros(shape=(h,w))
    theta = np.zeros(shape=(h,w))
    for i in range(h):
        for j in range(w):
            result[i][j] = ((img_1[i][j]**2 + img_2[i][j]**2)**(1/2))
            theta[i][j] = (math.atan2(img_2[i][j], img_1[i][j]))*180/math.pi
            if theta[i][j] < 0:
                theta[i][j] = -theta[i][j]
    return result, theta
```

Result Comparison



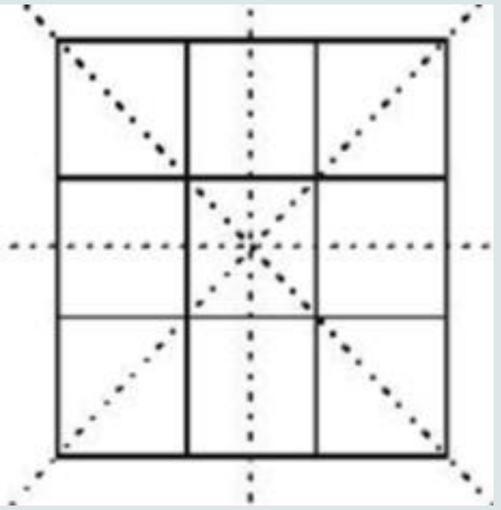
04. Non Maximum Suppression

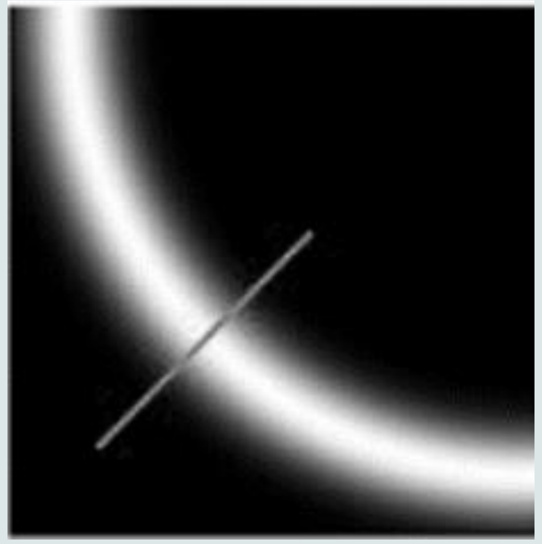
1. First Approach
2. Second Approach
3. Result Comparison

1) First Approach

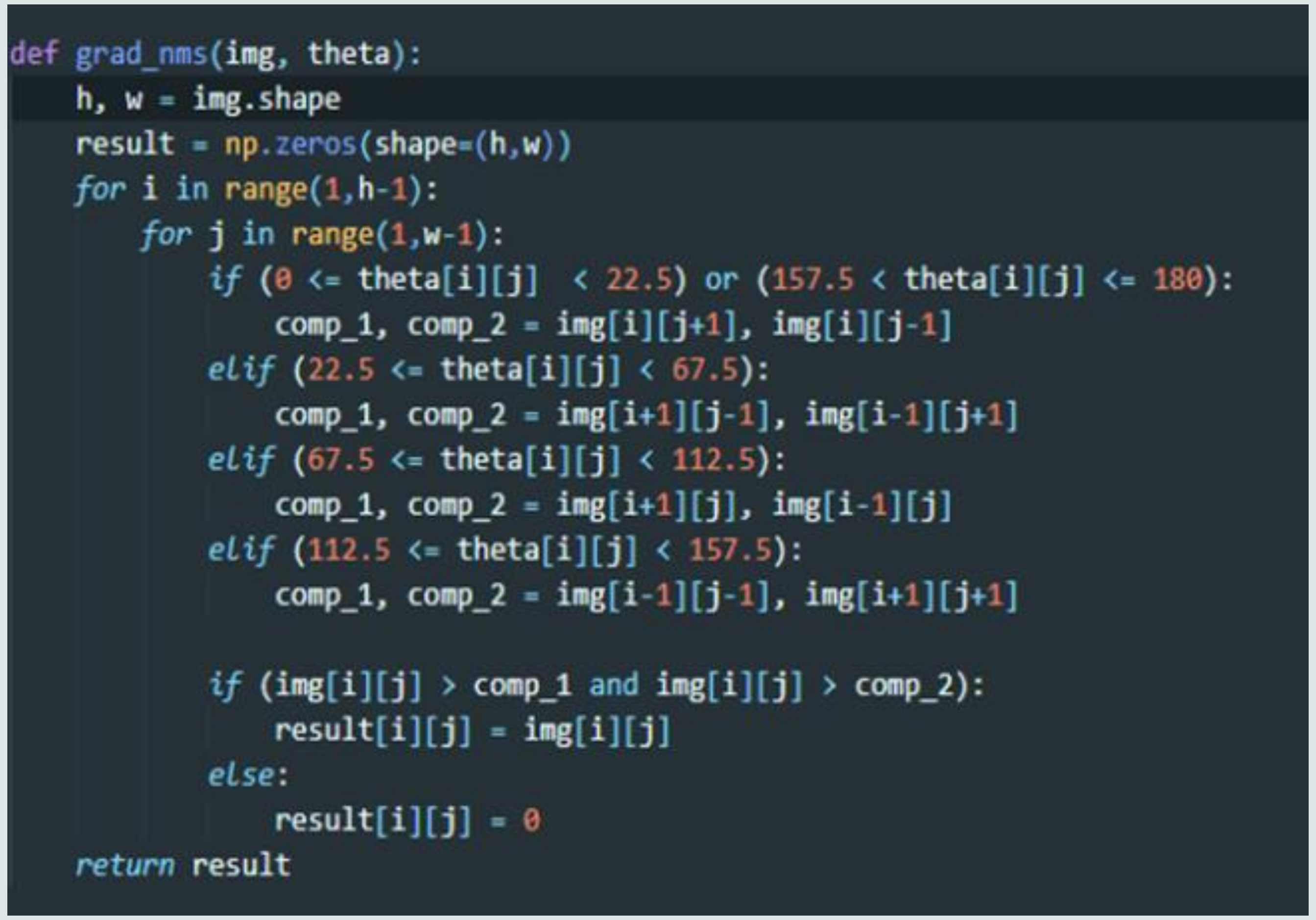
Low based cut off suppression

- gradient를 이용해 해당 방향과 이웃하는 두 픽셀과 비교하는 방식





-rounded gradient = 0	compare with image[i+1] [j], image[i-1] [j]
-rounded gradient = 45	compare with image[i+1] [j-1], image[i-1] [j+1]
-rounded gradient = 90	compare with image[i] [j-1], image[i] [j+1]
-rounded gradient = 135	compare with image[i-1] [j-1], image[i+1] [j+1]

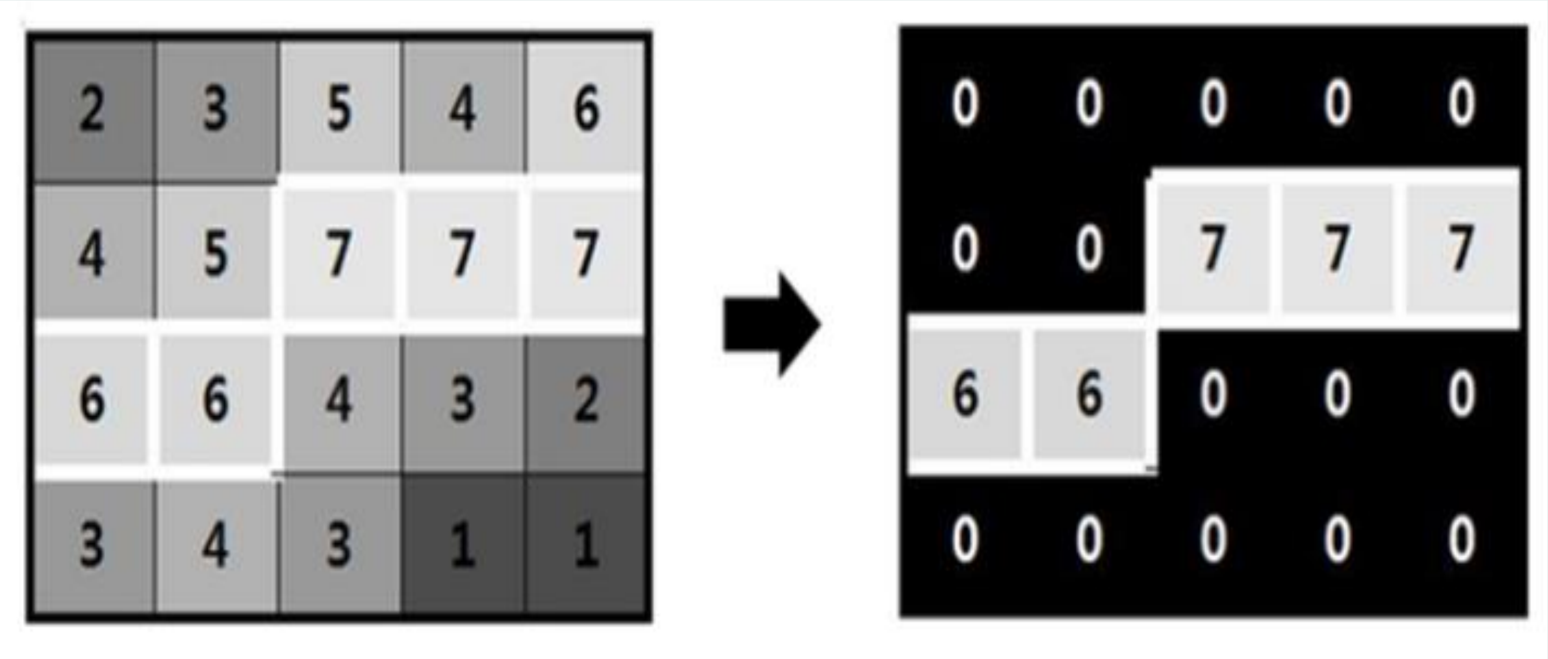


```python
def grad_nms(img, theta):
    h, w = img.shape
    result = np.zeros(shape=(h,w))
    for i in range(1,h-1):
        for j in range(1,w-1):
            if (0 <= theta[i][j]  < 22.5) or (157.5 < theta[i][j] <= 180):
                comp_1, comp_2 = img[i][j+1], img[i][j-1]
            elif (22.5 <= theta[i][j] < 67.5):
                comp_1, comp_2 = img[i+1][j-1], img[i-1][j+1]
            elif (67.5 <= theta[i][j] < 112.5):
                comp_1, comp_2 = img[i+1][j], img[i-1][j]
            elif (112.5 <= theta[i][j] < 157.5):
                comp_1, comp_2 = img[i-1][j-1], img[i+1][j+1]

            if (img[i][j] > comp_1 and img[i][j] > comp_2):
                result[i][j] = img[i][j]
            else:
                result[i][j] = 0
    return result
```

2) Second Approach

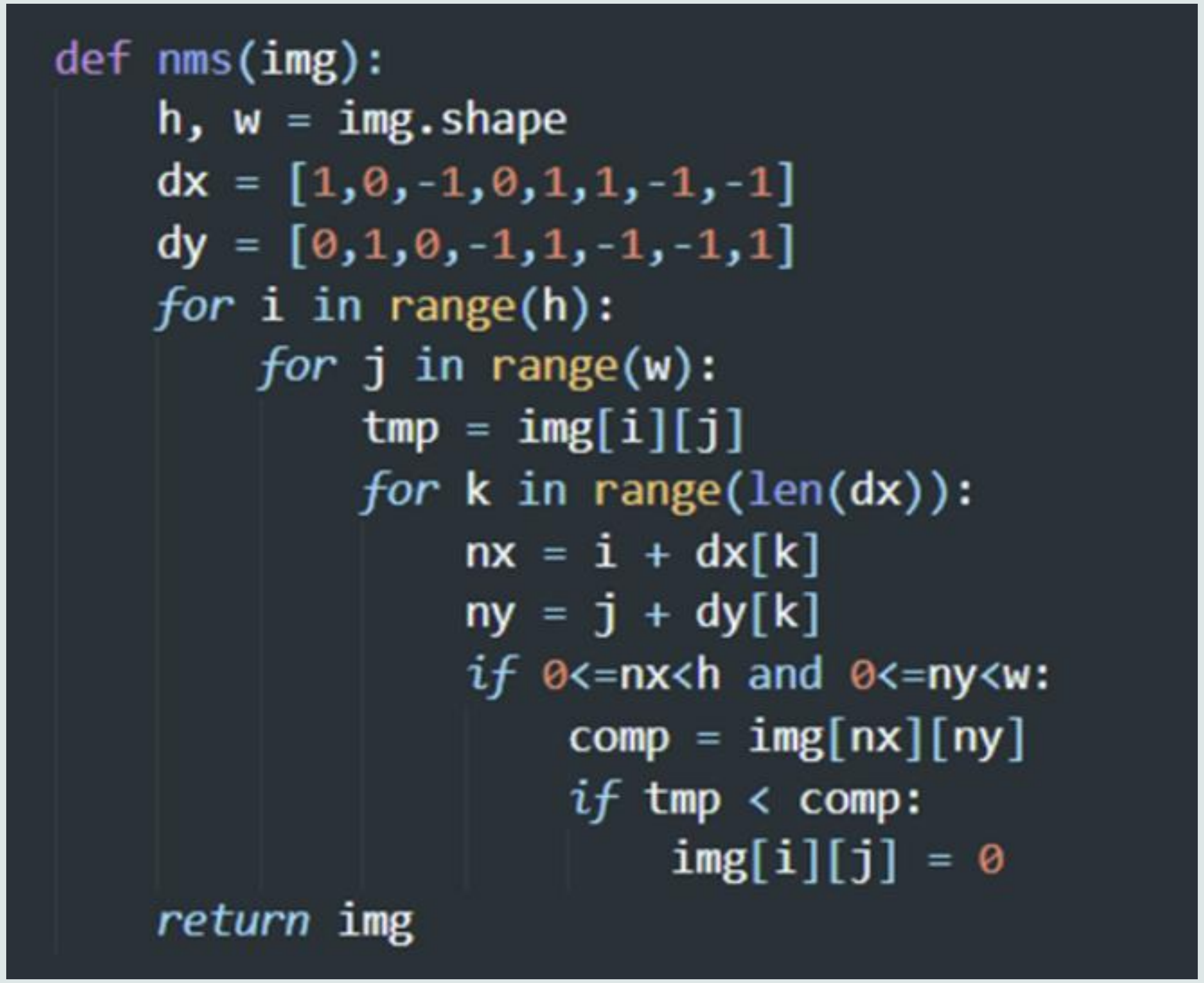
8 neighborhood

- 중심 픽셀을 기준으로 인접한 픽셀 값과 비교



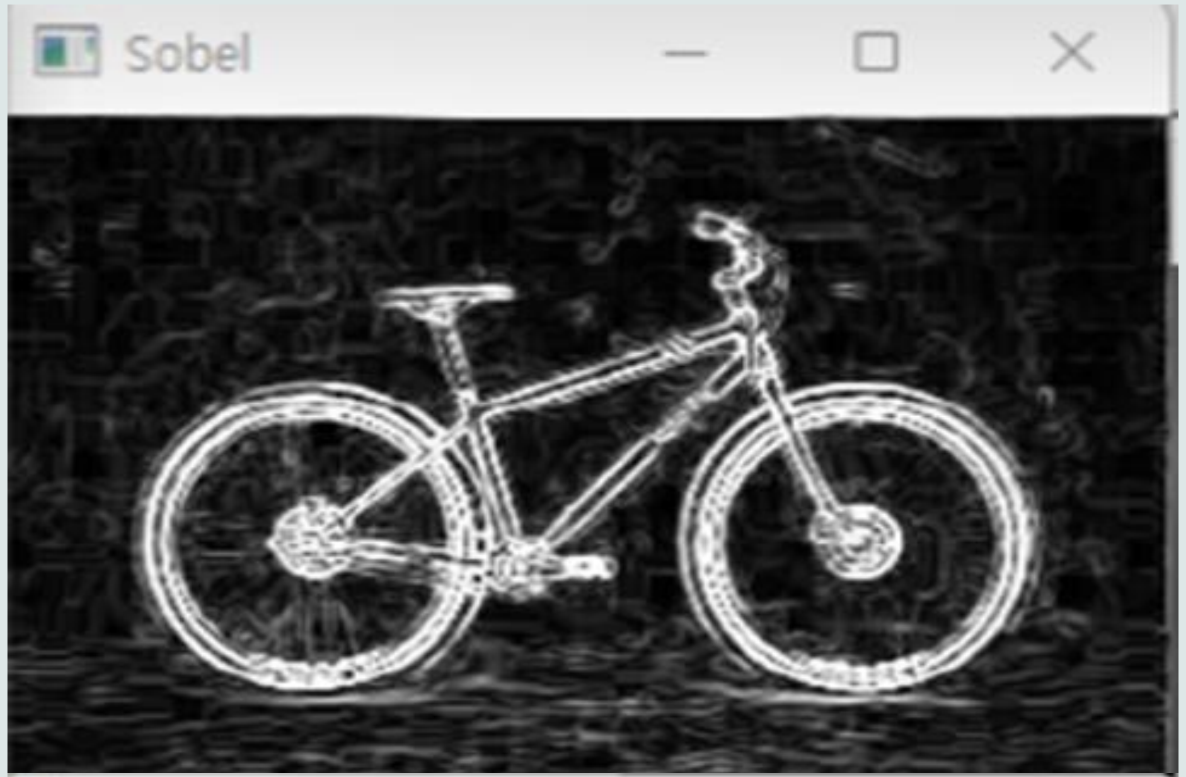
2	3	5	4	6
4	5	7	7	7
6	6	4	3	2
3	4	3	1	1

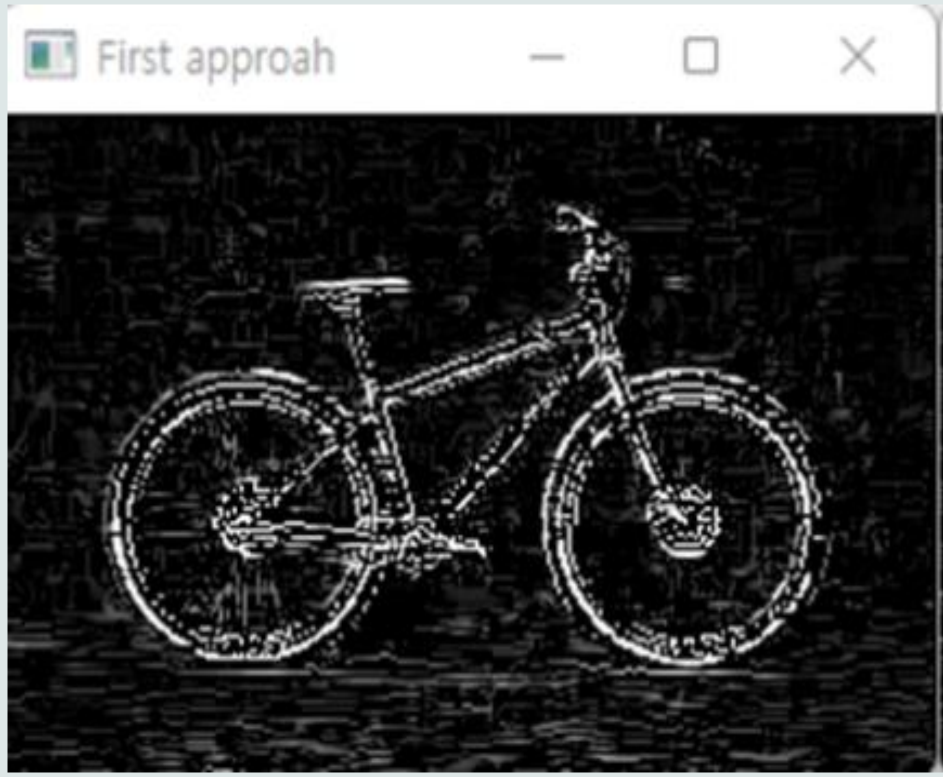
0	0	0	0	0
0	0	7	7	7
6	6	0	0	0
0	0	0	0	0

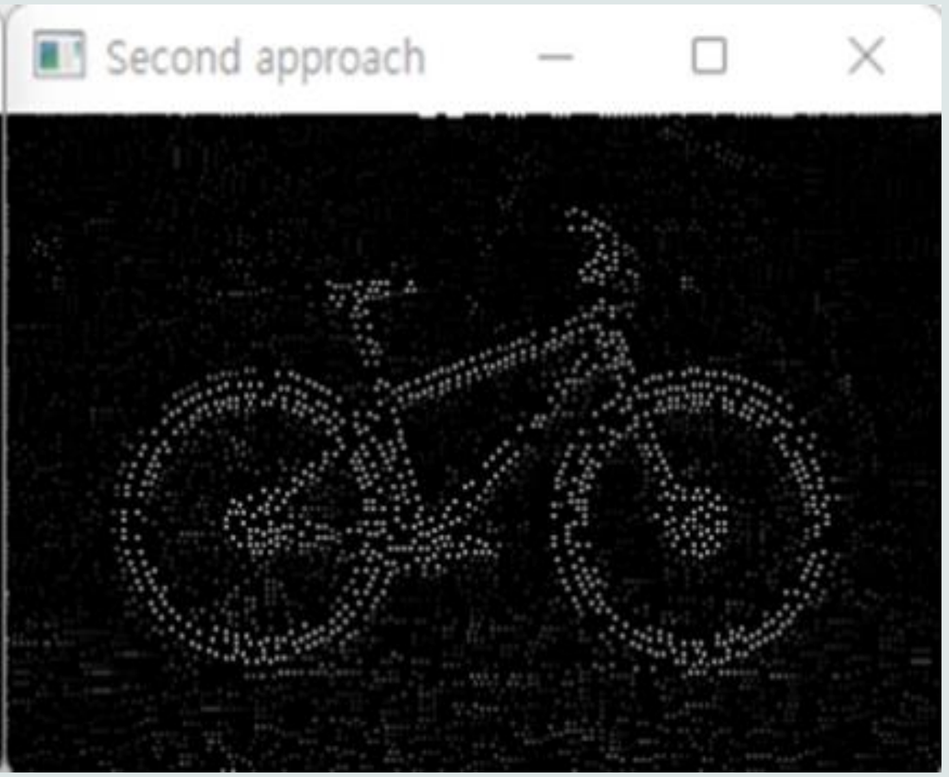


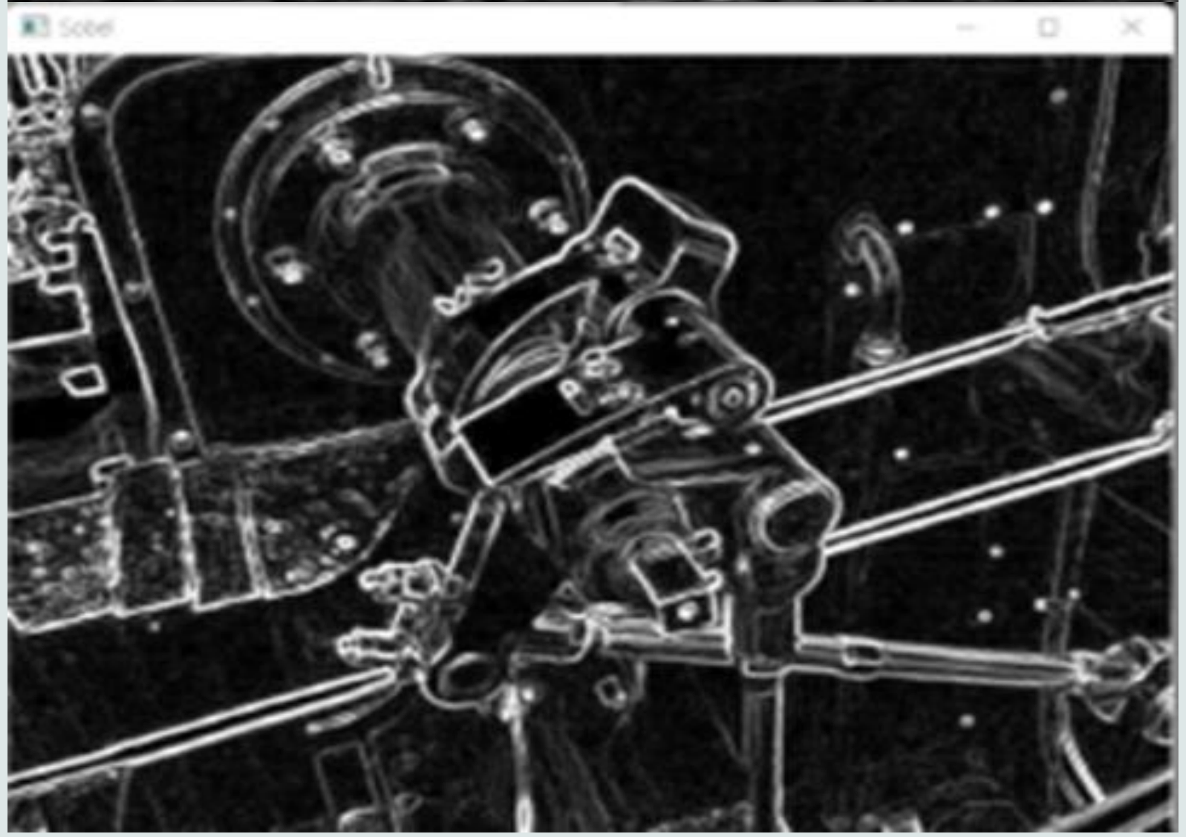
```python
def nms(img):
    h, w = img.shape
    dx = [1,0,-1,0,1,1,-1,-1]
    dy = [0,1,0,-1,1,-1,-1,1]
    for i in range(h):
        for j in range(w):
            tmp = img[i][j]
            for k in range(len(dx)):
                nx = i + dx[k]
                ny = j + dy[k]
                if 0<=nx<h and 0<=ny<w:
                    comp = img[nx][ny]
                    if tmp < comp:
                        img[i][j] = 0
    return img
```

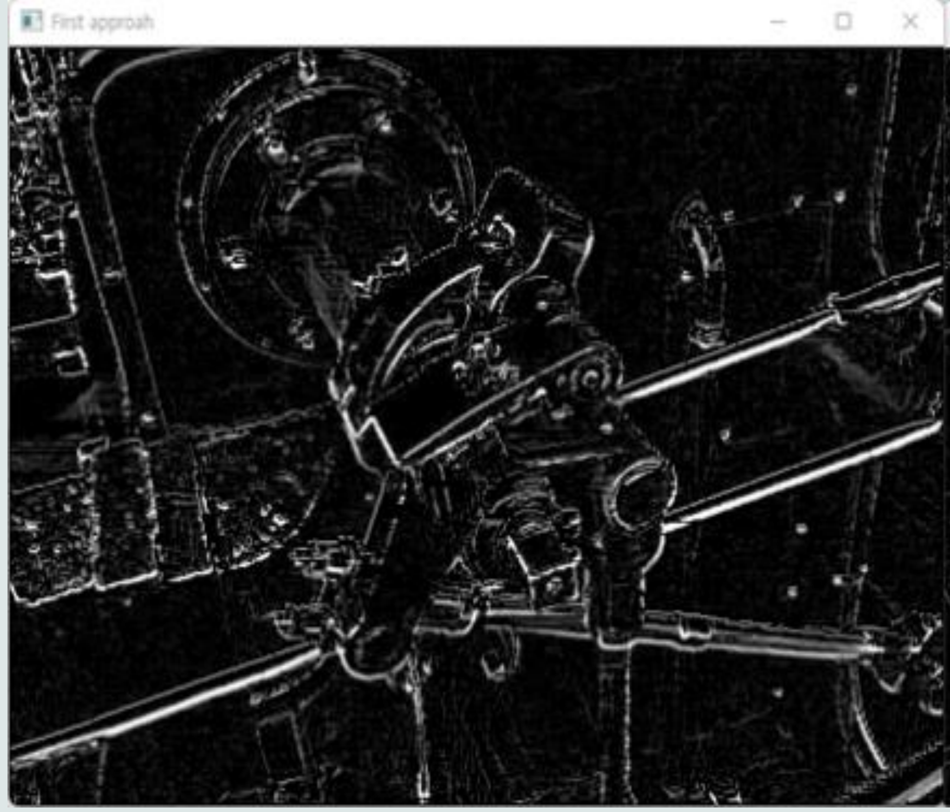
3) Result Comparison

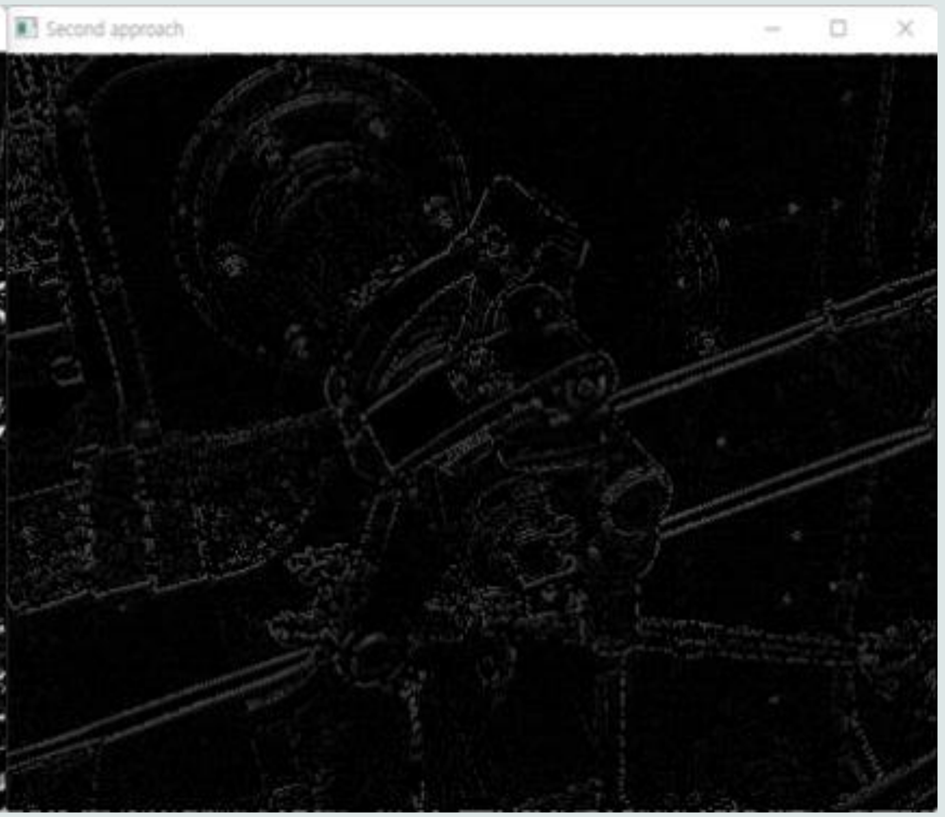






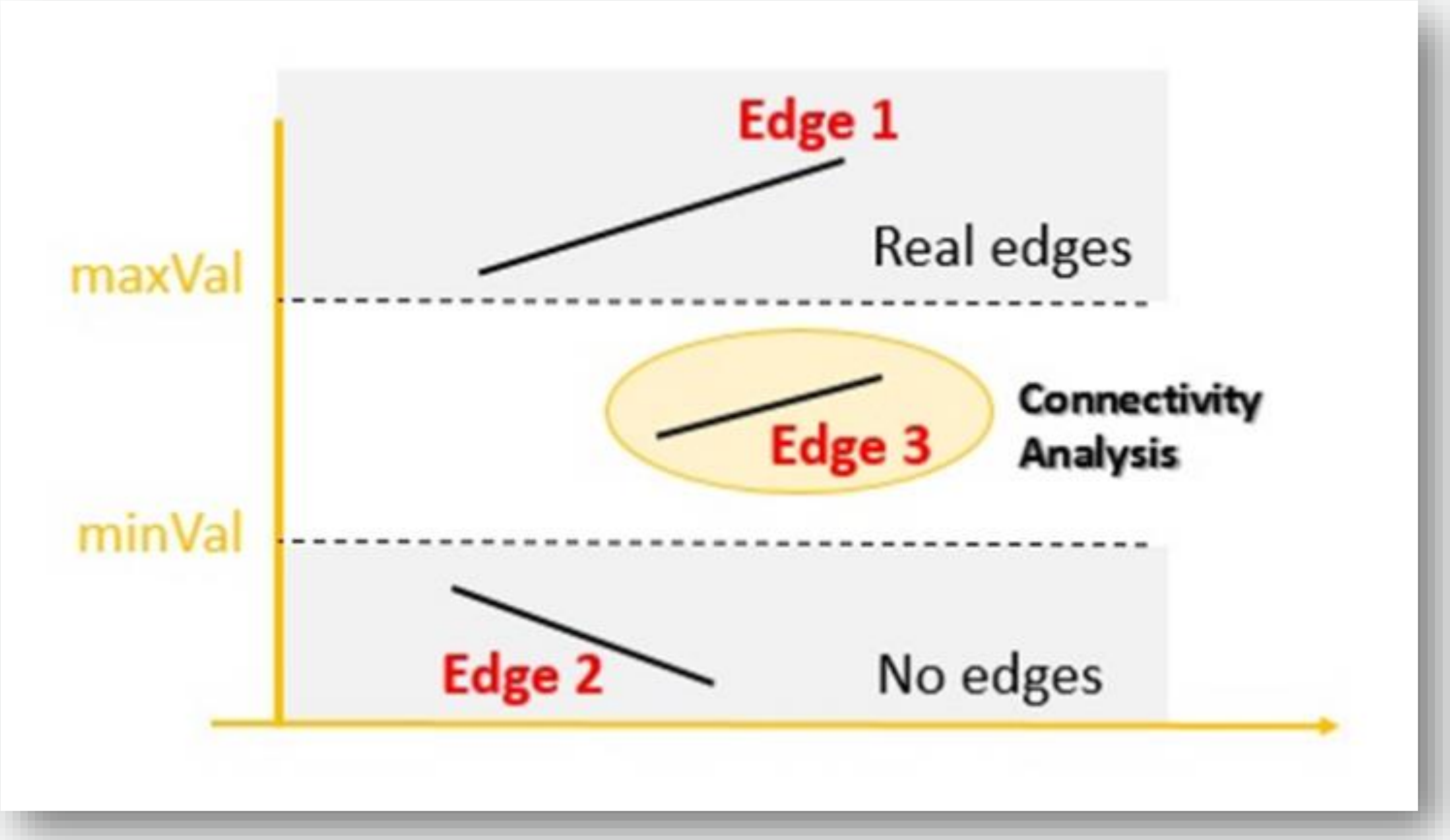






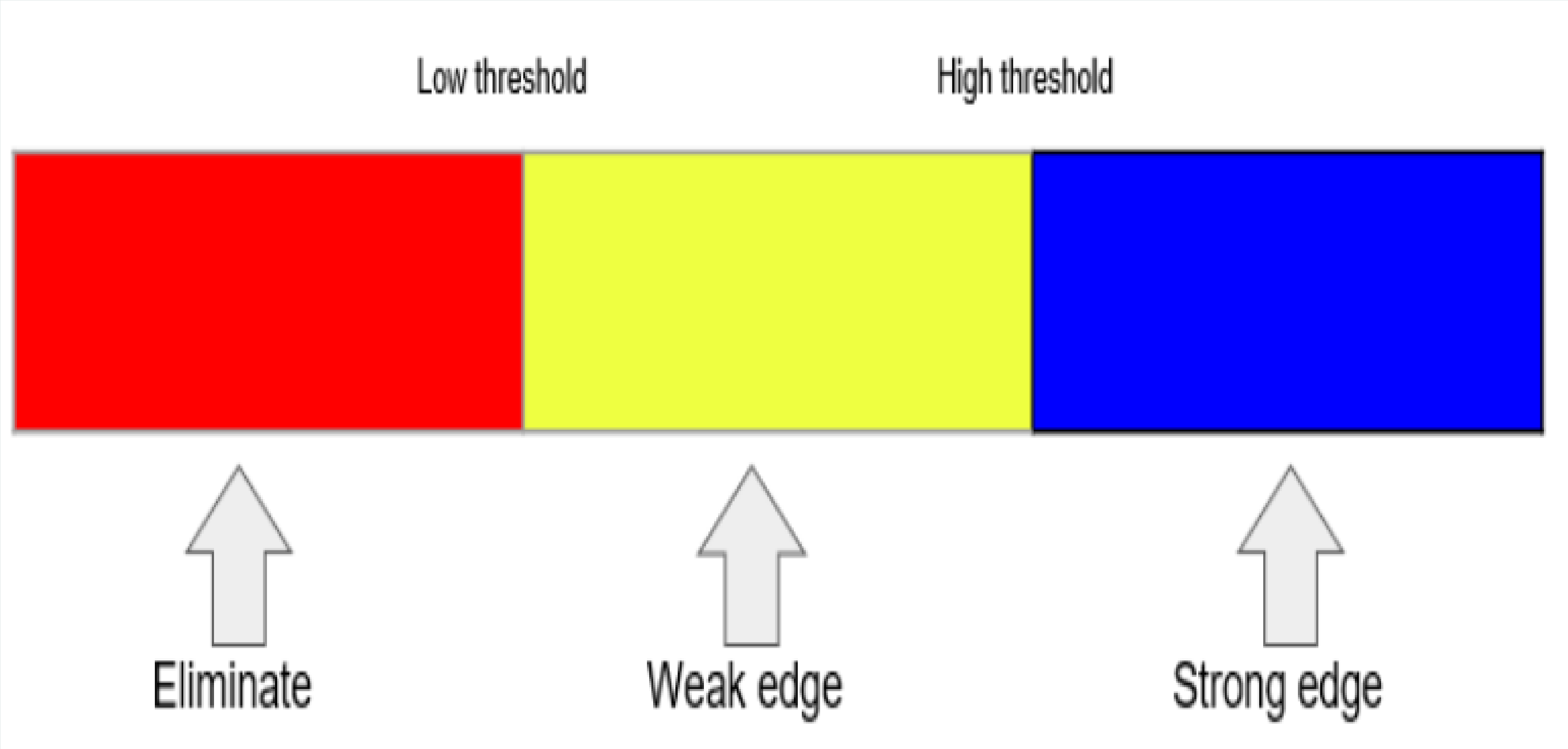
05. Hysteresis Thresholding

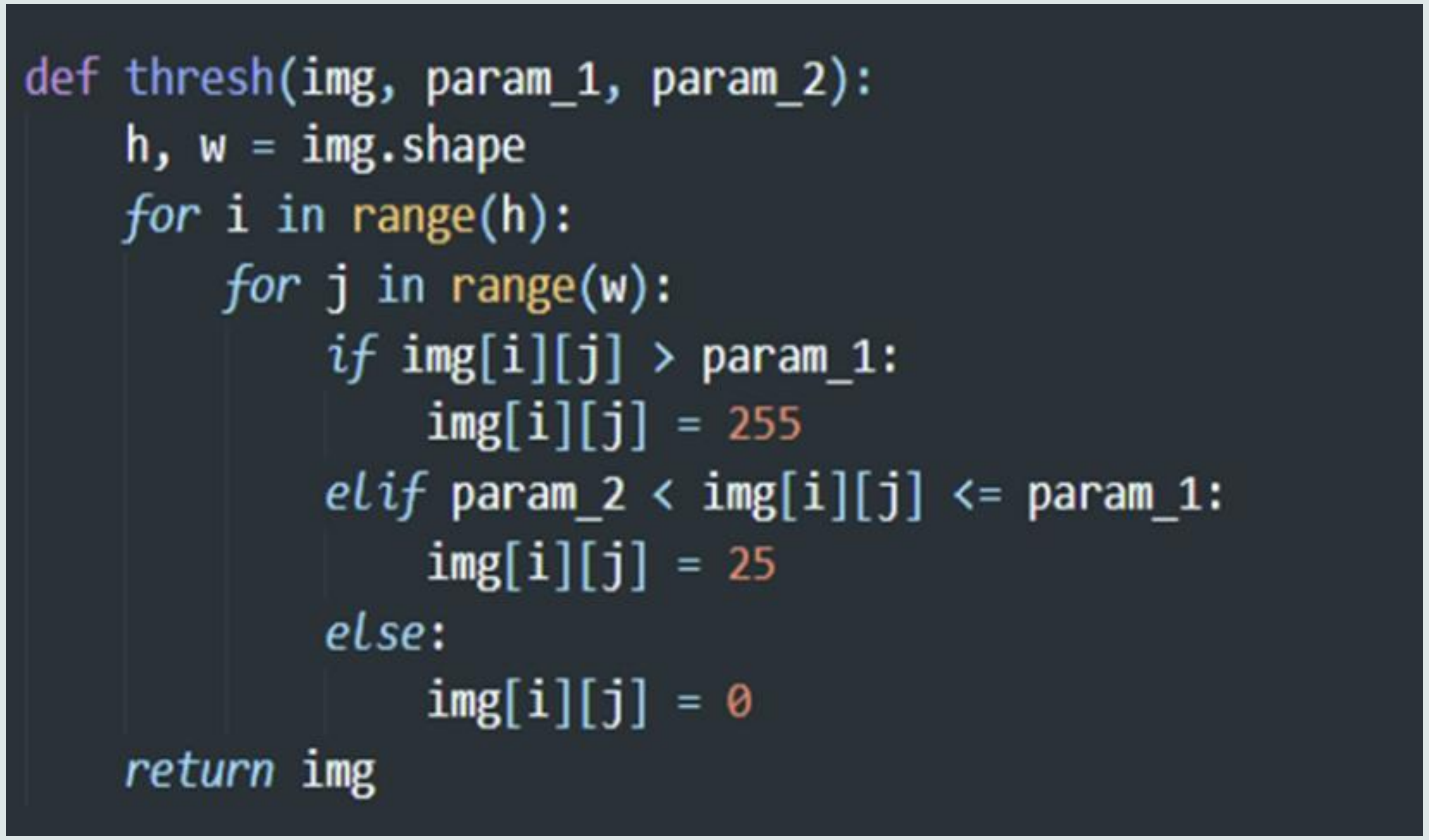
두 개의 경계 값 high threshold, low threshold를 지정해서 경계 영역에 있는 픽셀들 중 high threshold 밖의 픽셀과 연결성이 없는 픽셀을 제거하는 알고리즘



1. Double Threshold
2. Edge Tracking by Hysteresis
3. Result

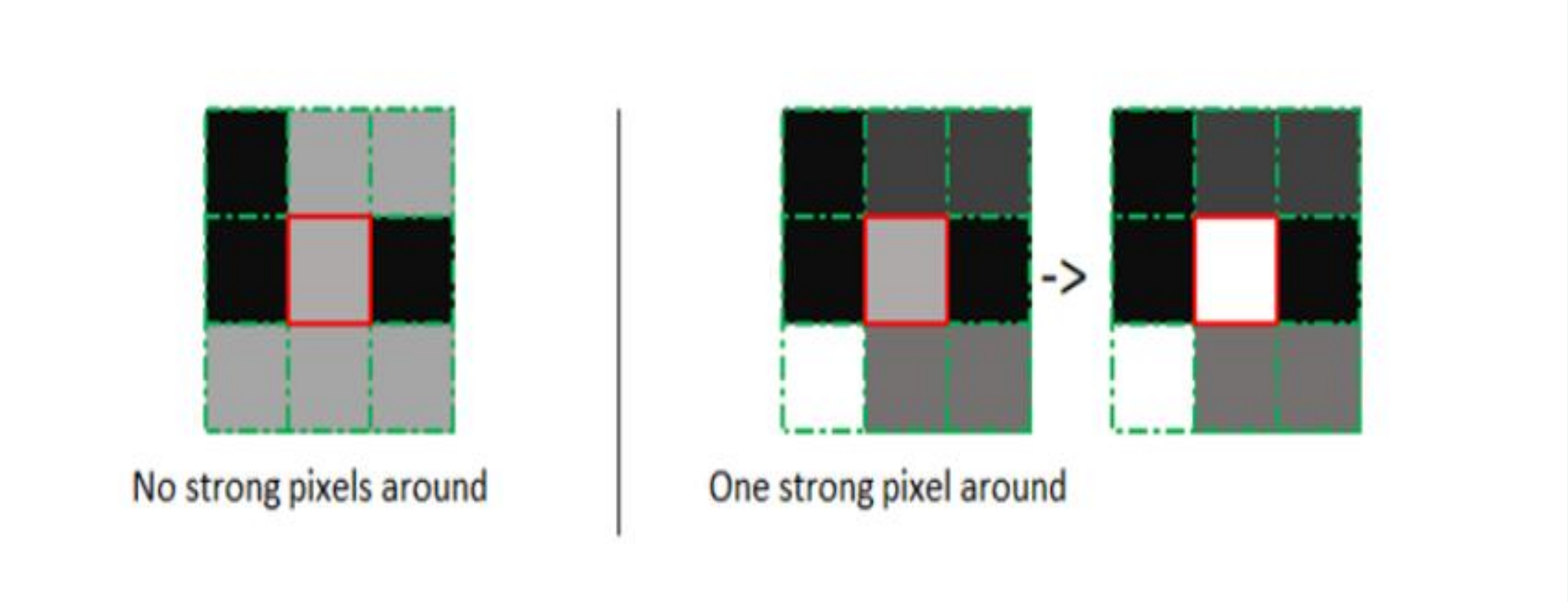
1) Double Threshold





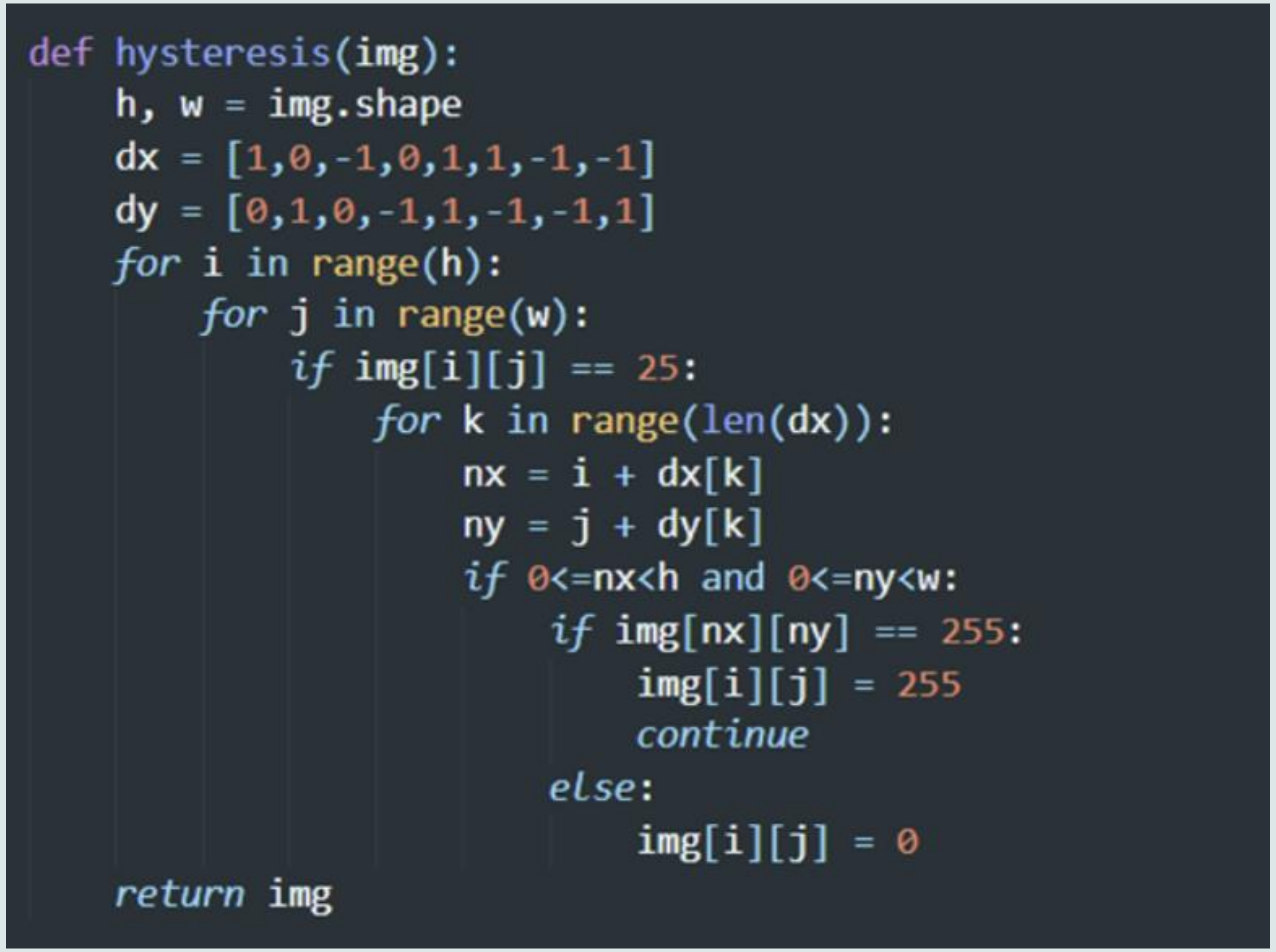
```
def thresh(img, param_1, param_2):
    h, w = img.shape
    for i in range(h):
        for j in range(w):
            if img[i][j] > param_1:
                img[i][j] = 255
            elif param_2 < img[i][j] <= param_1:
                img[i][j] = 25
            else:
                img[i][j] = 0
    return img
```

2) Edge Tracking by Hysteresis



No strong pixels around

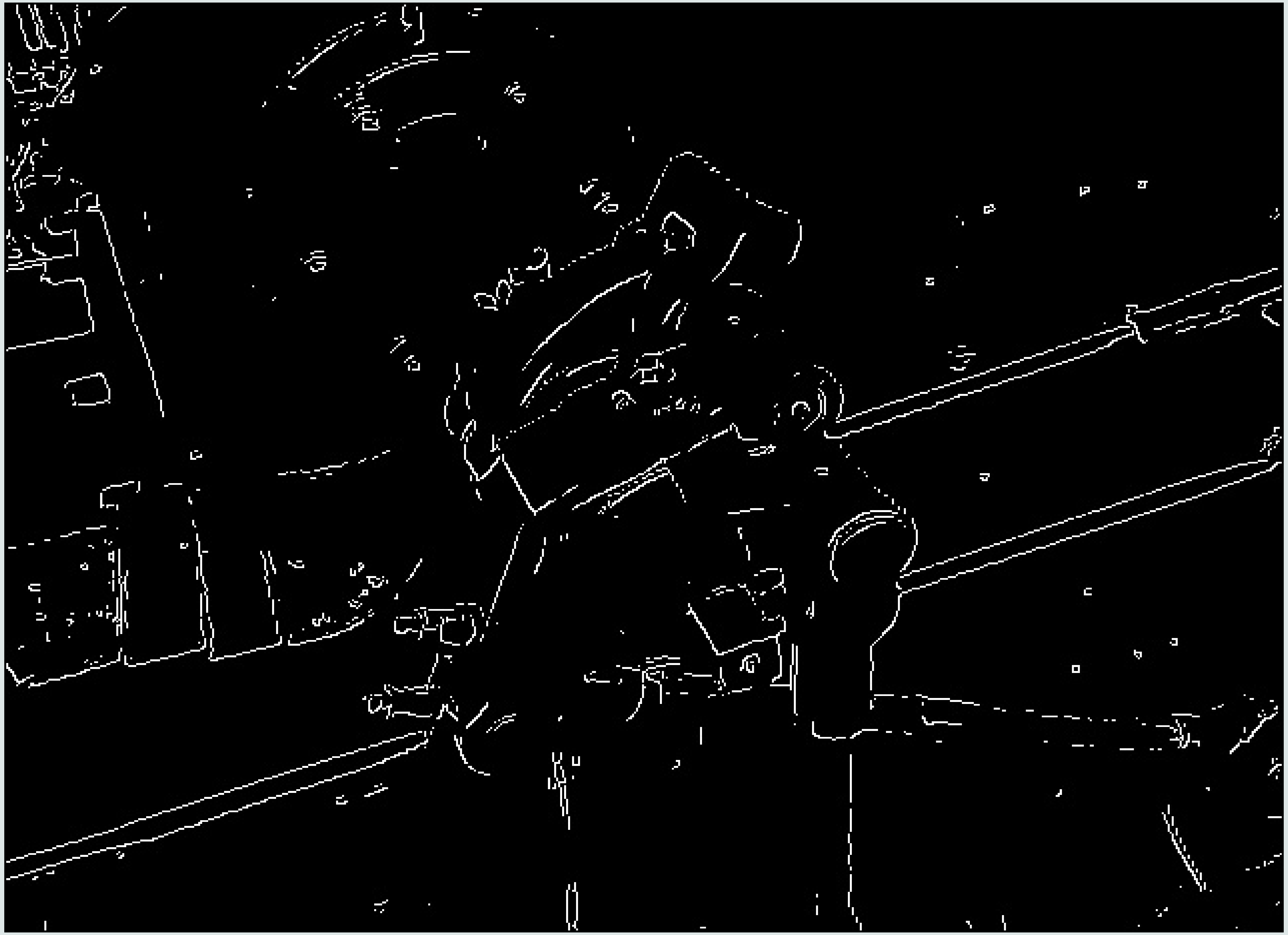
One strong pixel around



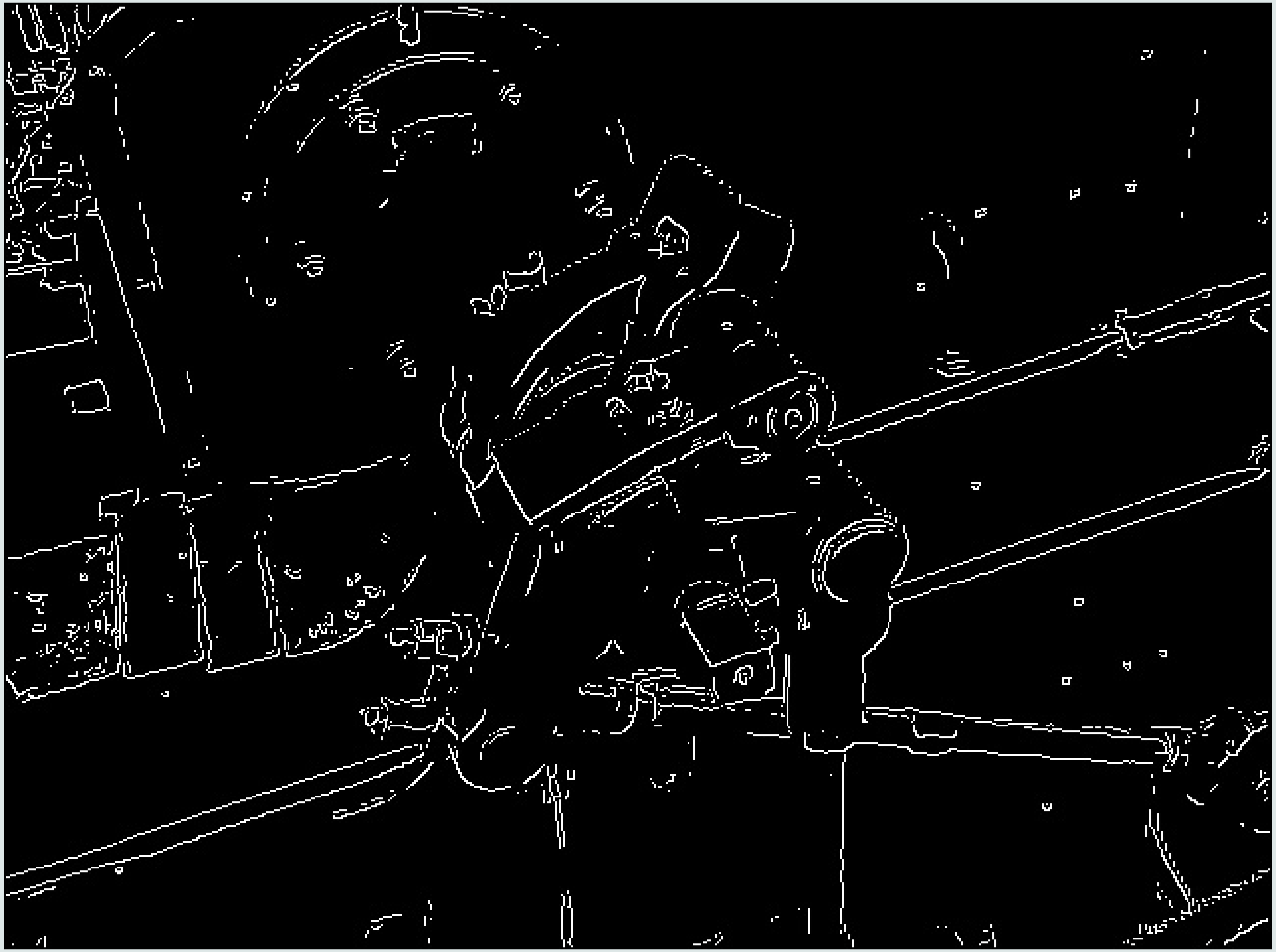
```python
def hysteresis(img):
    h, w = img.shape
    dx = [1,0,-1,0,1,1,-1,-1]
    dy = [0,1,0,-1,1,-1,-1,1]
    for i in range(h):
        for j in range(w):
            if img[i][j] == 25:
                for k in range(len(dx)):
                    nx = i + dx[k]
                    ny = j + dy[k]
                    if 0<=nx<h and 0<=ny<w:
                        if img[nx][ny] == 255:
                            img[i][j] = 255
                            continue
                    else:
                        img[i][j] = 0
    return img
```

3) Result

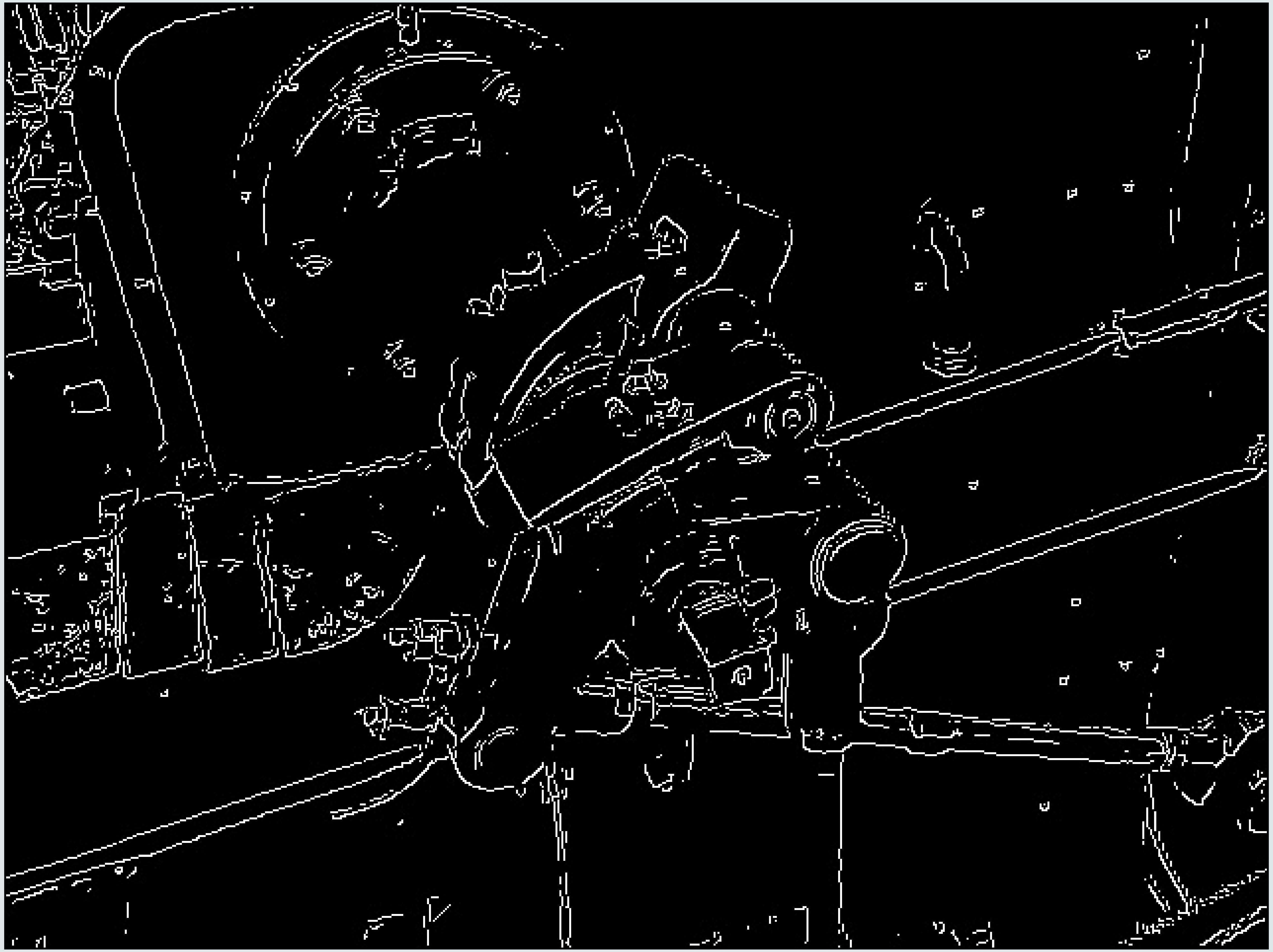
High : 200 Low : 50



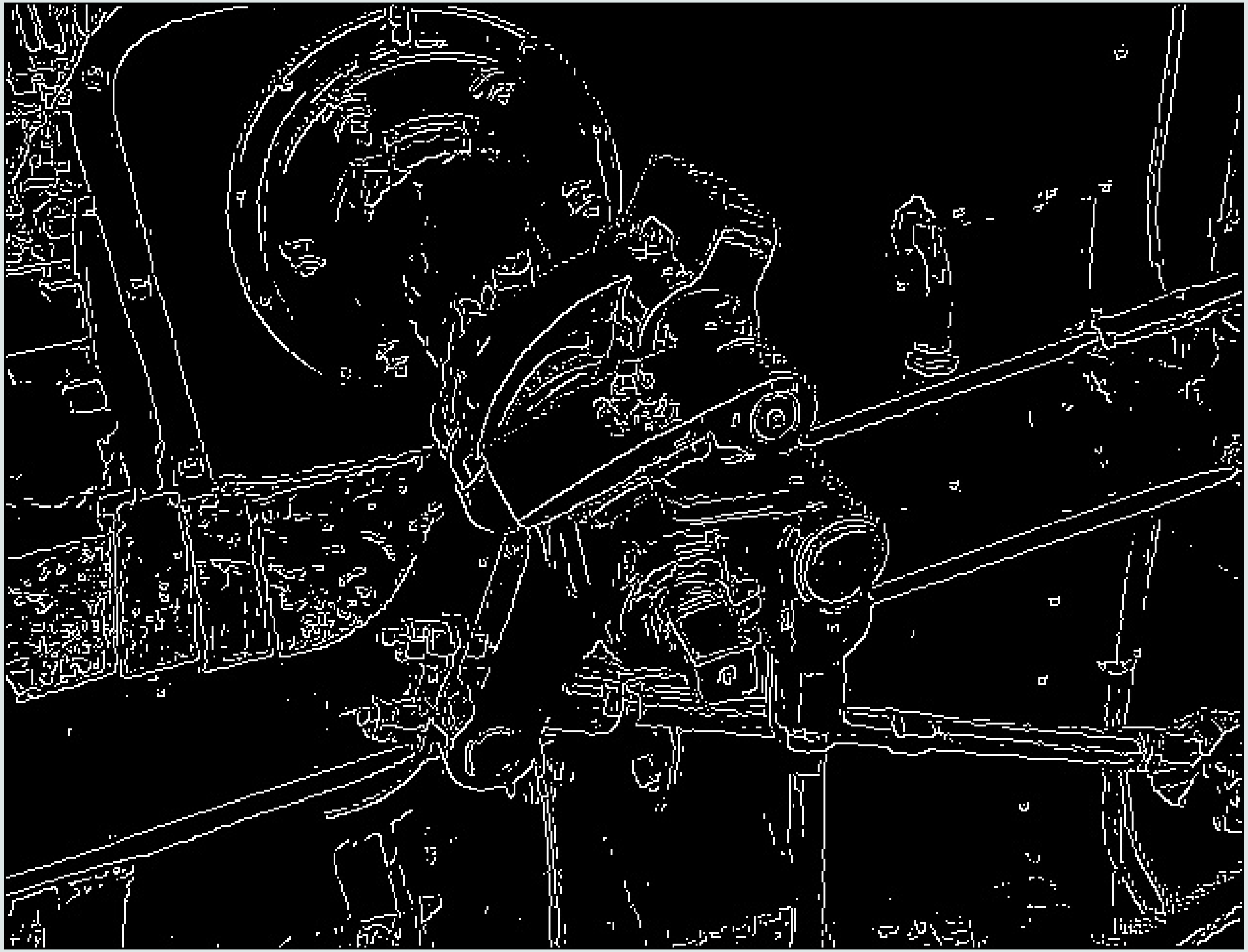
High : 150 Low : 50



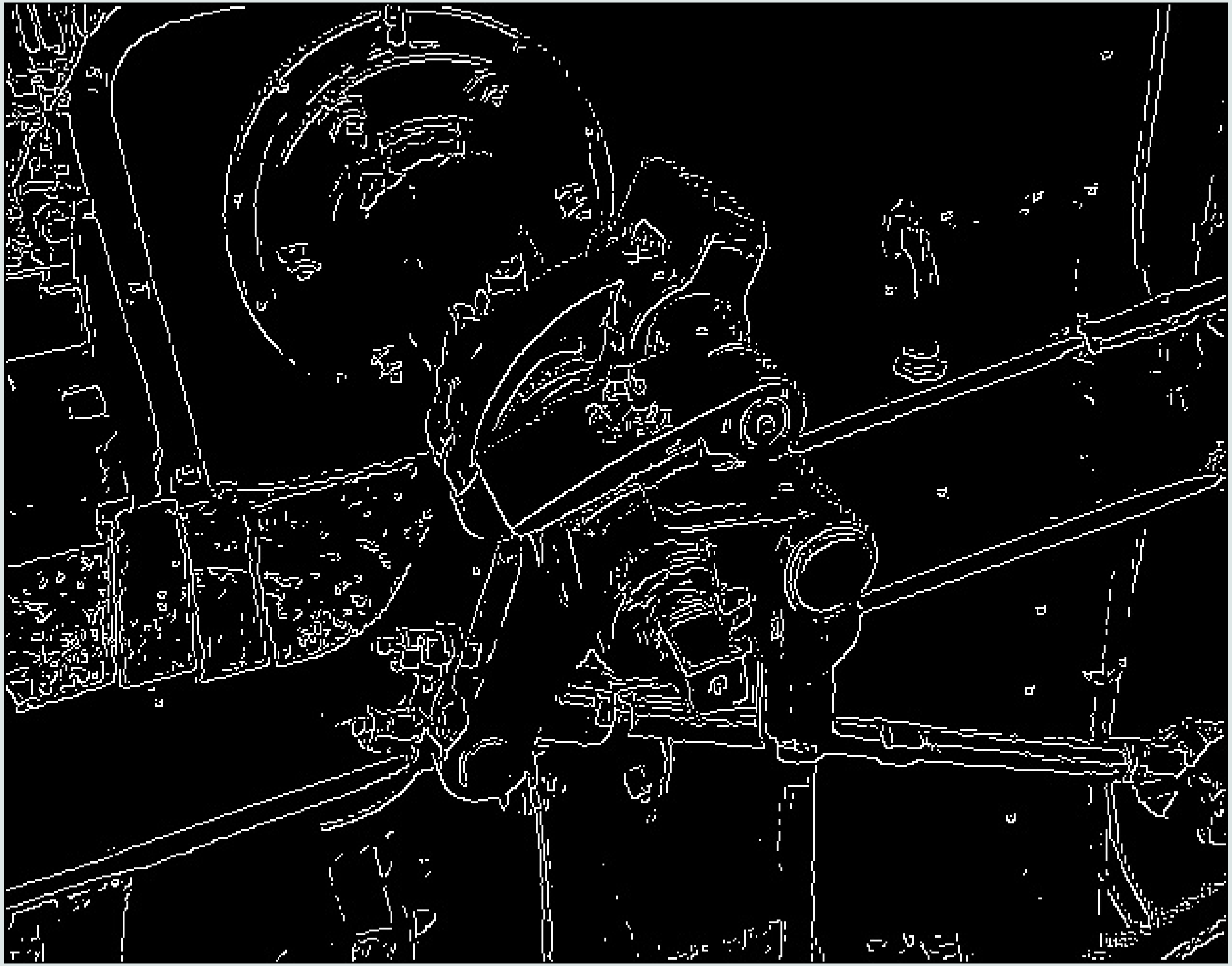
High : 120 Low : 50



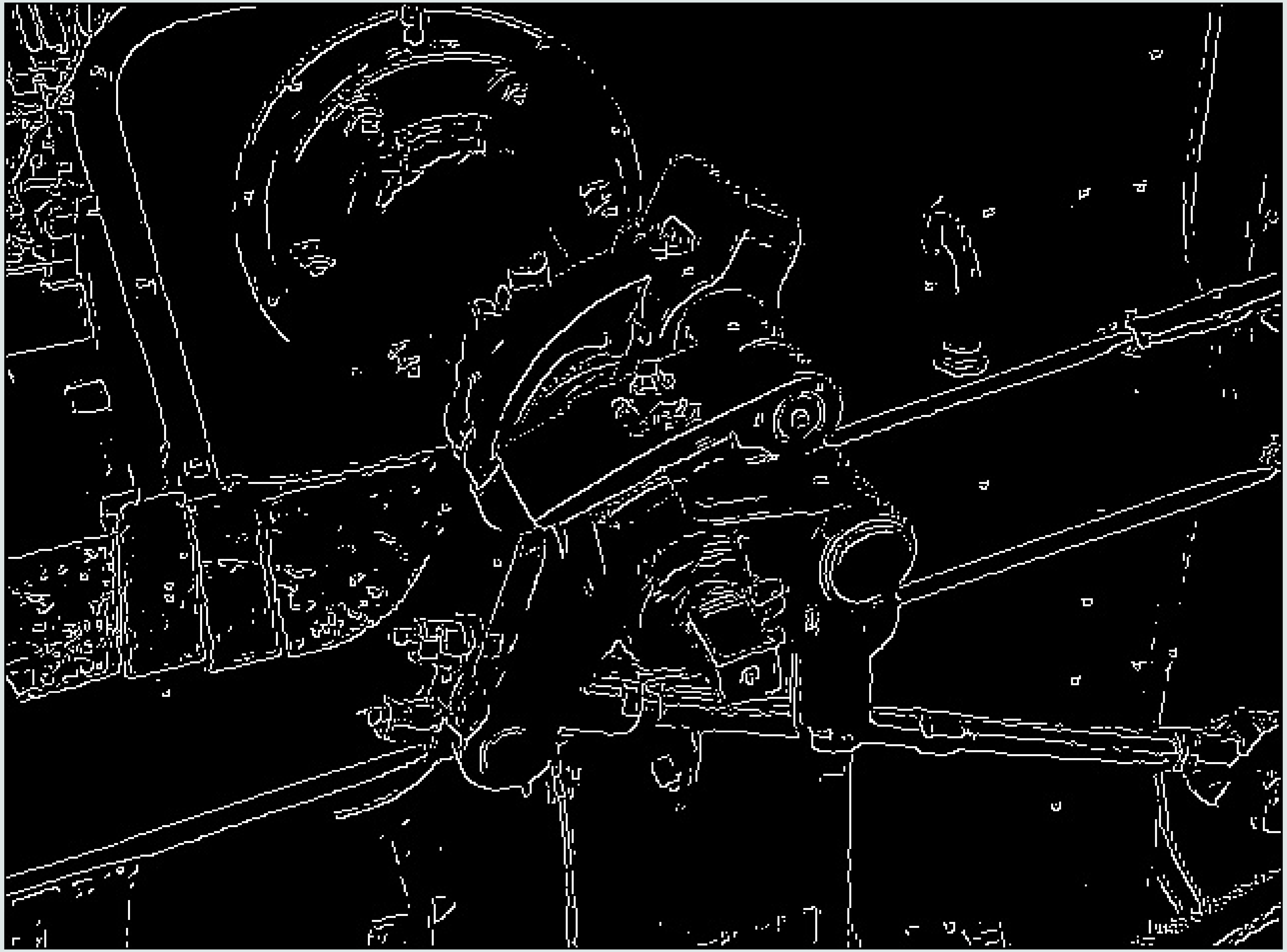
High : 70 Low : 50



High : 85 Low : 50



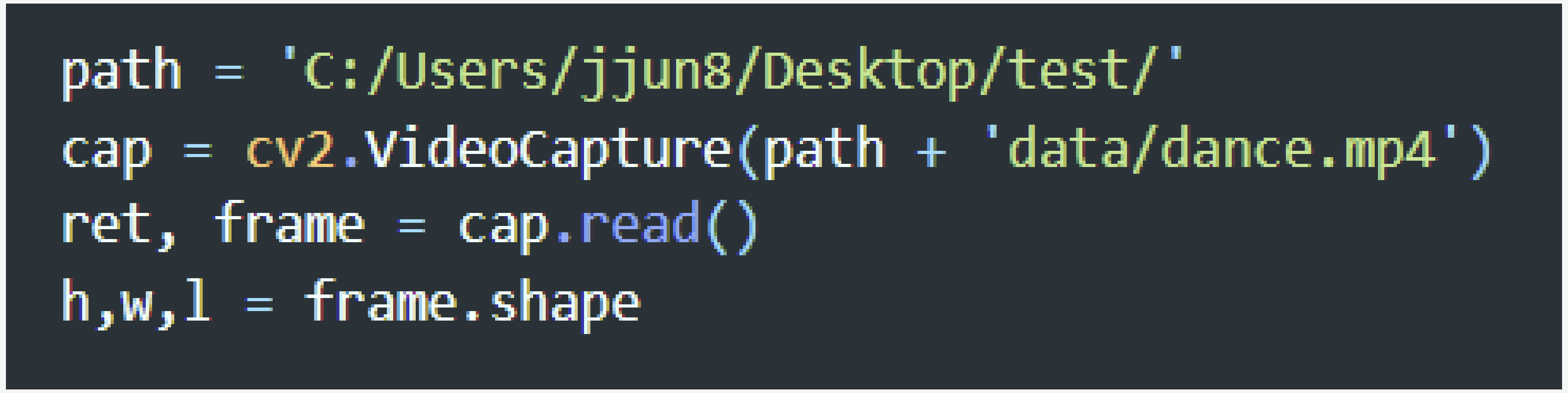
High : 100   Low : 50



06. Experiment

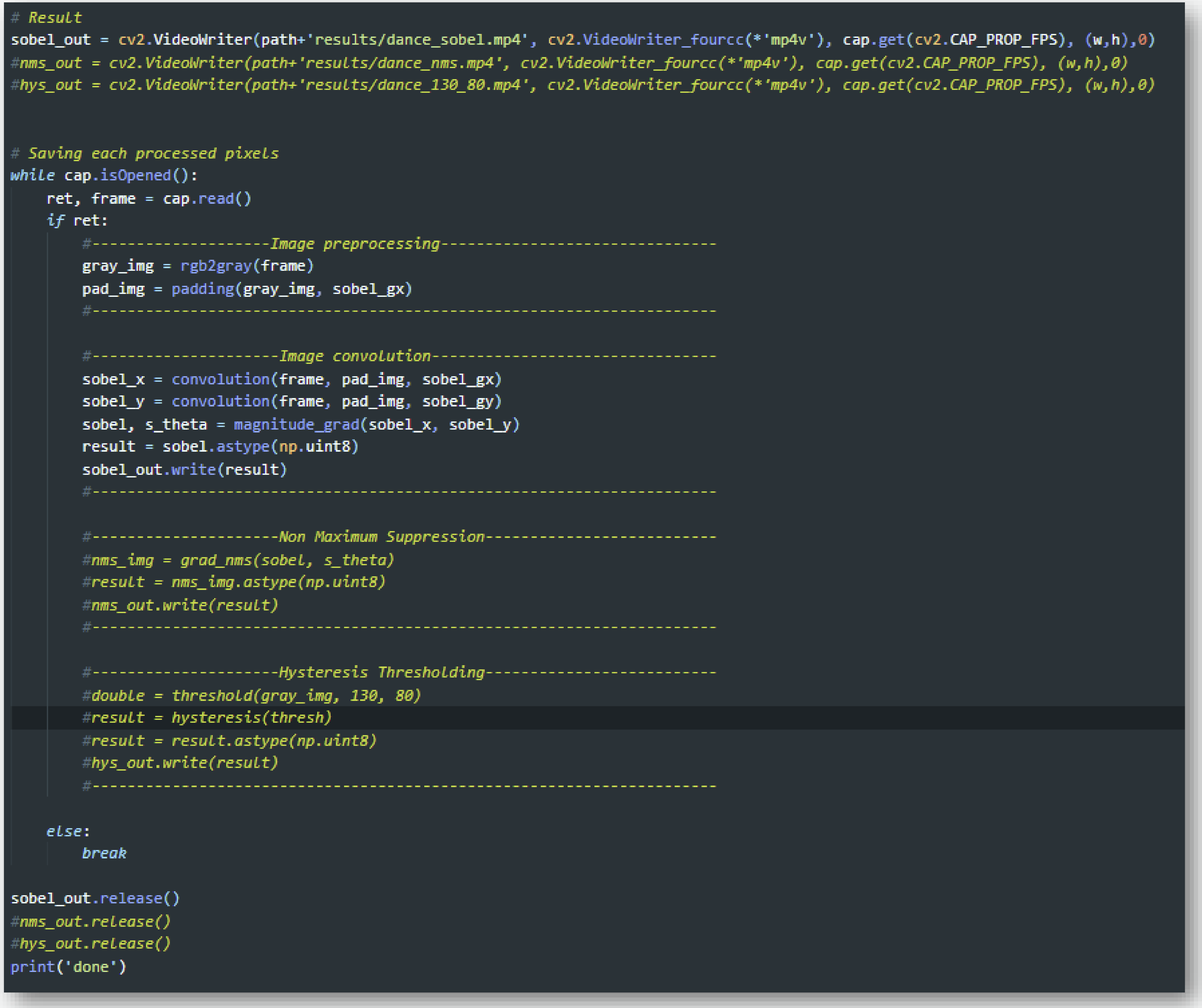
1. First Video
2. Second Video

Experiment-1

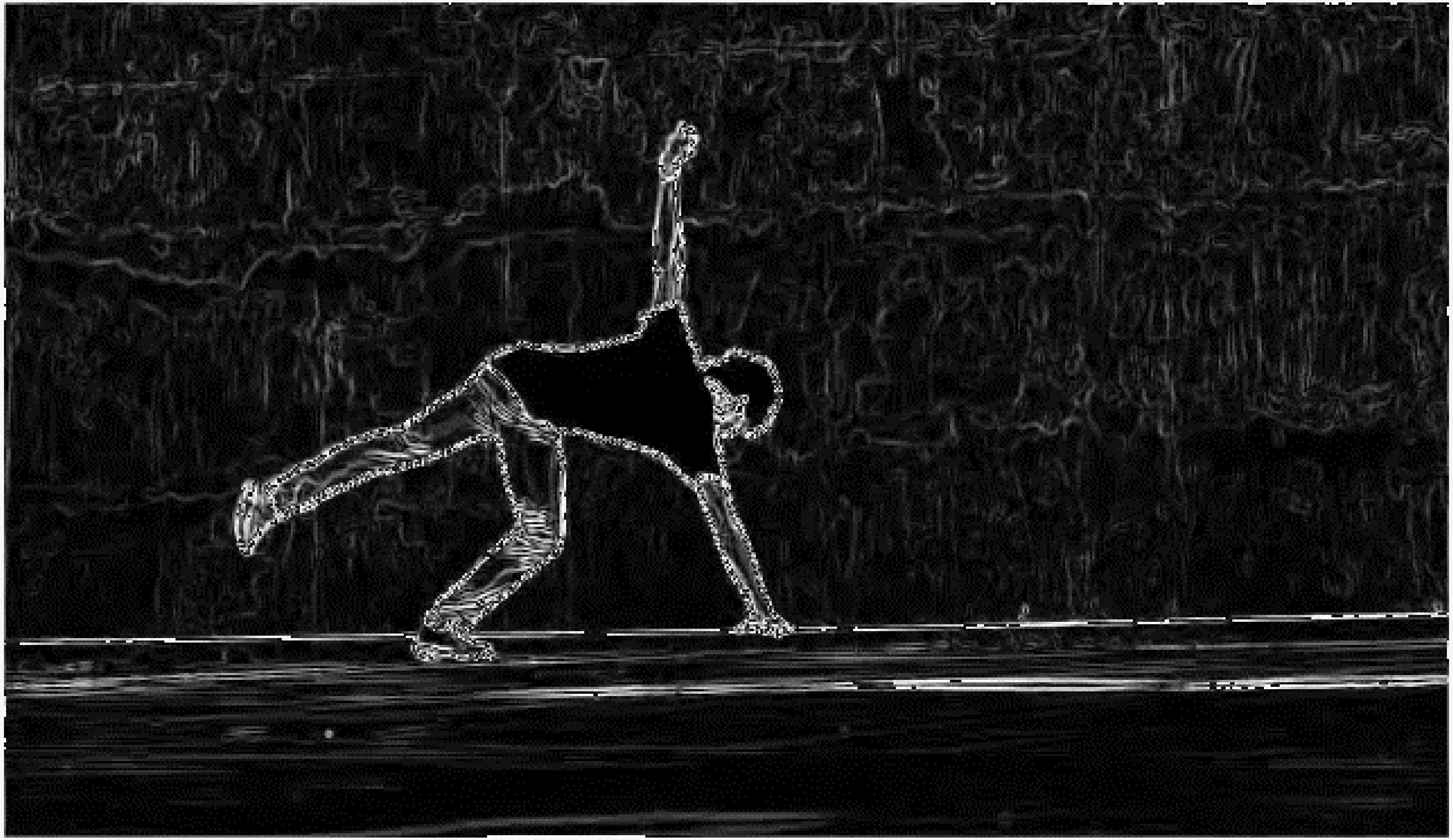


```
path = 'C:/Users/jjun8/Desktop/test/'
cap = cv2.VideoCapture(path + 'data/dance.mp4')
ret, frame = cap.read()
h,w,l = frame.shape
```

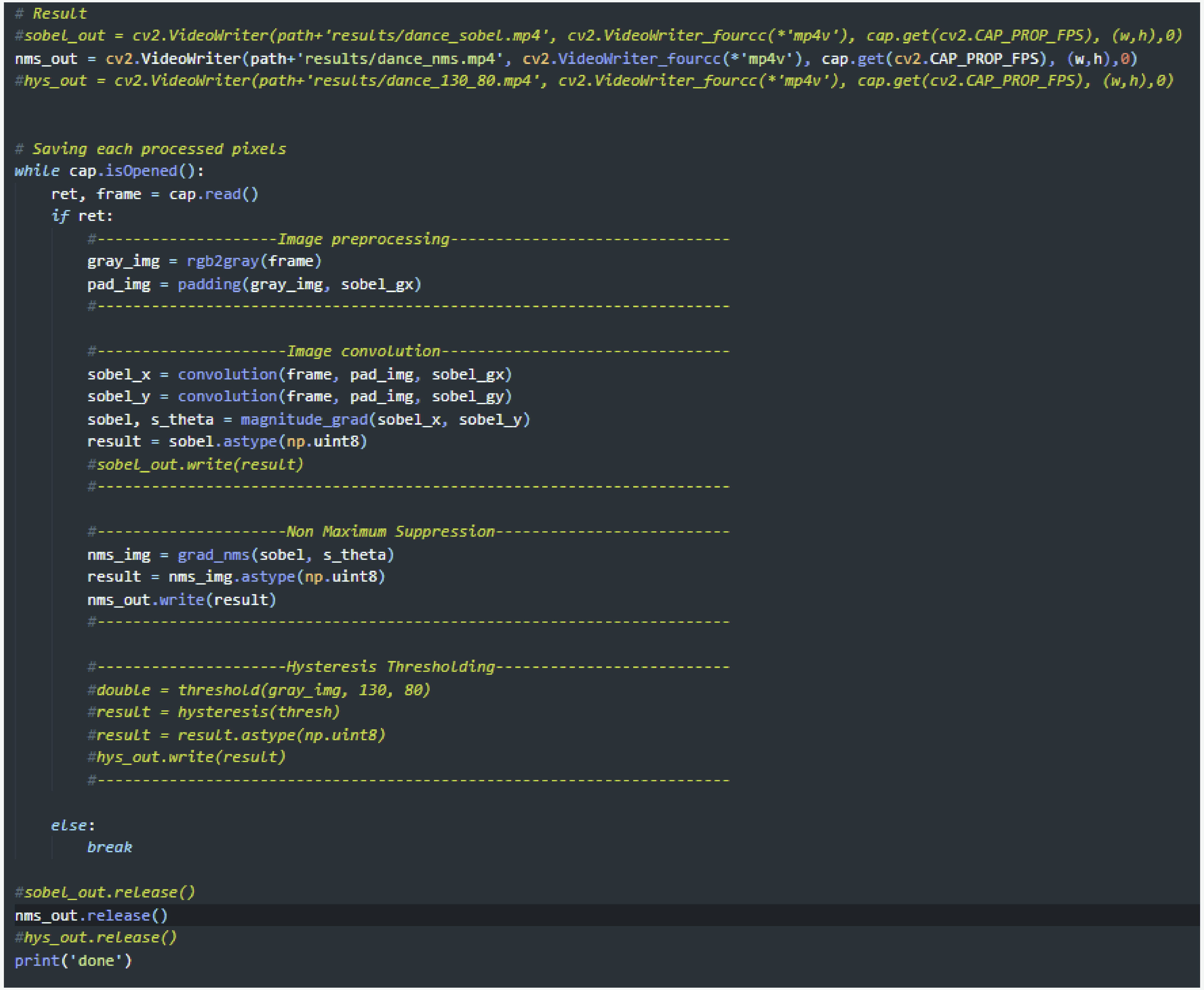
Sobel mask



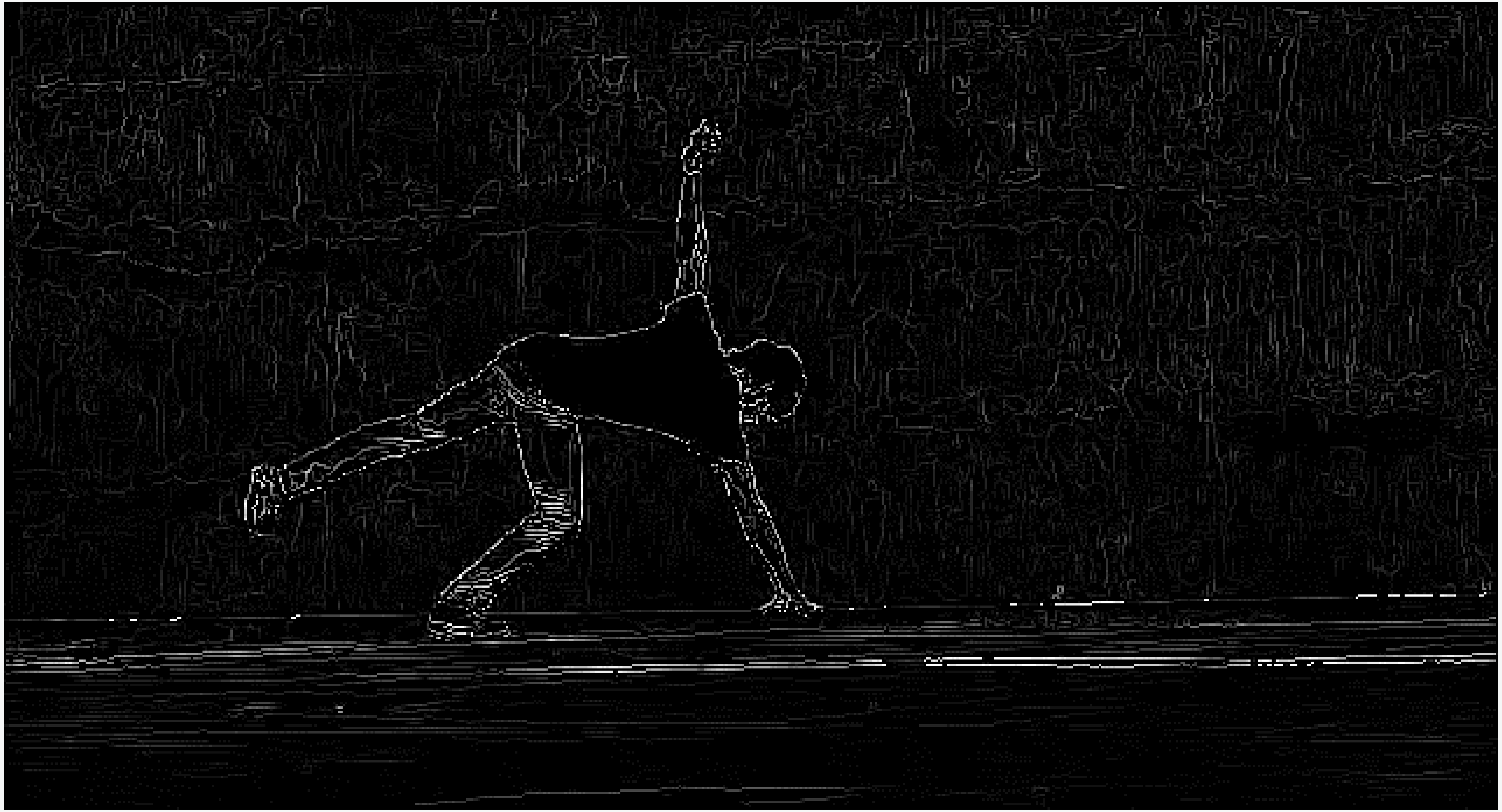
```
# Result
sobel_out = cv2.VideoWriter(path+'results/dance_sobel.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
#nms_out = cv2.VideoWriter(path+'results/dance_nms.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
#hys_out = cv2.VideoWriter(path+'results/dance_130_80.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)


# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        #--------------------Image preprocessing-------------------------------
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img, sobel_gx)
        #----------------------------------------------------------------------

        #---------------------Image convolution--------------------------------
        sobel_x = convolution(frame, pad_img, sobel_gx)
        sobel_y = convolution(frame, pad_img, sobel_gy)
        sobel, s_theta = magnitude_grad(sobel_x, sobel_y)
        result = sobel.astype(np.uint8)
        sobel_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Non Maximum Suppression--------------------------
        #nms_img = grad_nms(sobel, s_theta)
        #result = nms_img.astype(np.uint8)
        #nms_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Hysteresis Thresholding--------------------------
        #double = threshold(gray_img, 130, 80)
        #result = hysteresis(thresh)
        #result = result.astype(np.uint8)
        #hys_out.write(result)
        #----------------------------------------------------------------------

    else:
        break

sobel_out.release()
#nms_out.release()
#hys_out.release()
print('done')
```



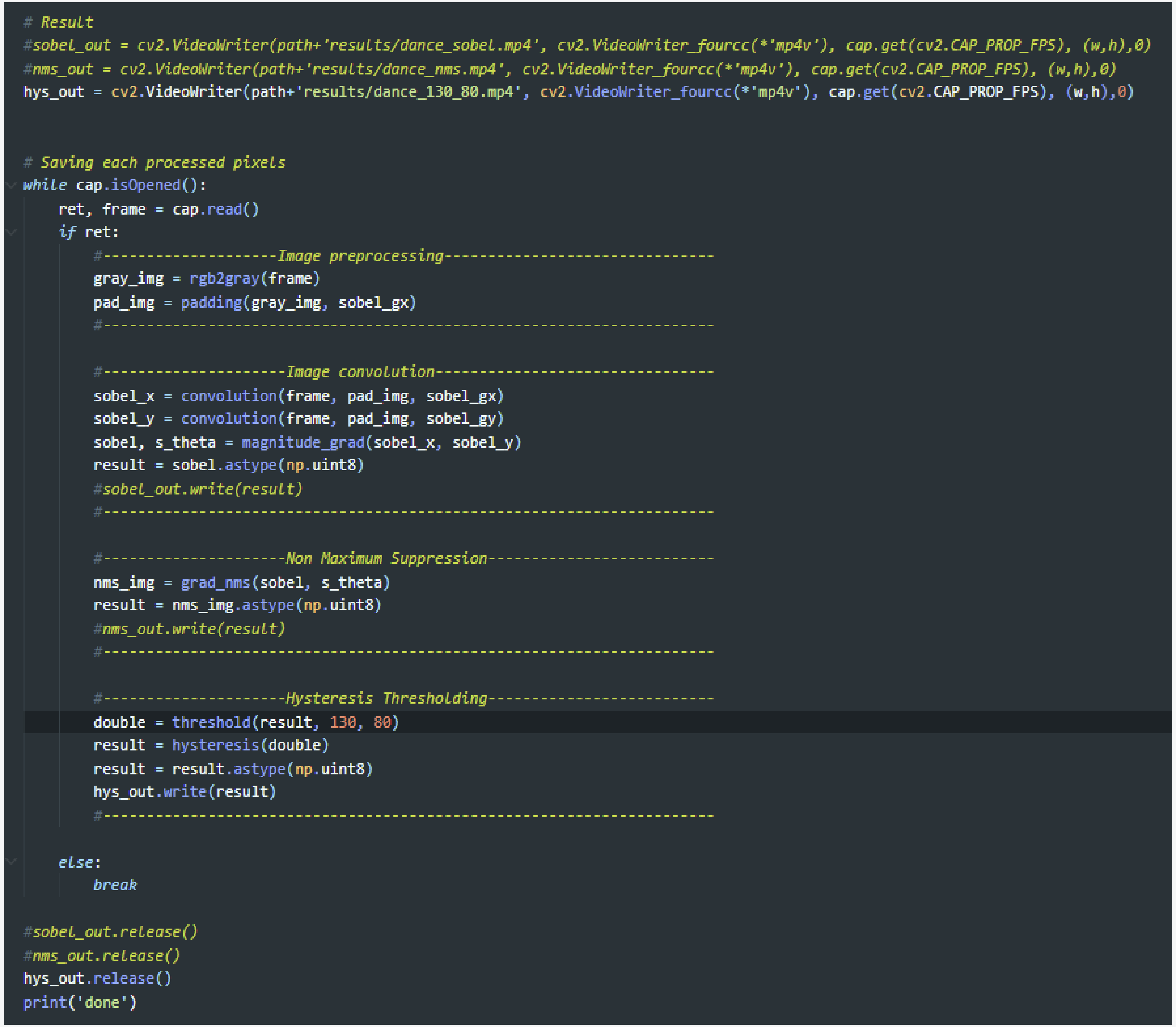
Non Maximum Suppression



```
# Result
#sobel_out = cv2.VideoWriter(path+'results/dance_sobel.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
nms_out = cv2.VideoWriter(path+'results/dance_nms.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
#hys_out = cv2.VideoWriter(path+'results/dance_130_80.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)


# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        #--------------------Image preprocessing-------------------------------
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img, sobel_gx)
        #----------------------------------------------------------------------

        #---------------------Image convolution--------------------------------
        sobel_x = convolution(frame, pad_img, sobel_gx)
        sobel_y = convolution(frame, pad_img, sobel_gy)
        sobel, s_theta = magnitude_grad(sobel_x, sobel_y)
        result = sobel.astype(np.uint8)
        #sobel_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Non Maximum Suppression--------------------------
        nms_img = grad_nms(sobel, s_theta)
        result = nms_img.astype(np.uint8)
        nms_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Hysteresis Thresholding--------------------------
        #double = threshold(gray_img, 130, 80)
        #result = hysteresis(thresh)
        #result = result.astype(np.uint8)
        #hys_out.write(result)
        #----------------------------------------------------------------------

    else:
        break

#sobel_out.release()
nms_out.release()
#hys_out.release()
print('done')
```

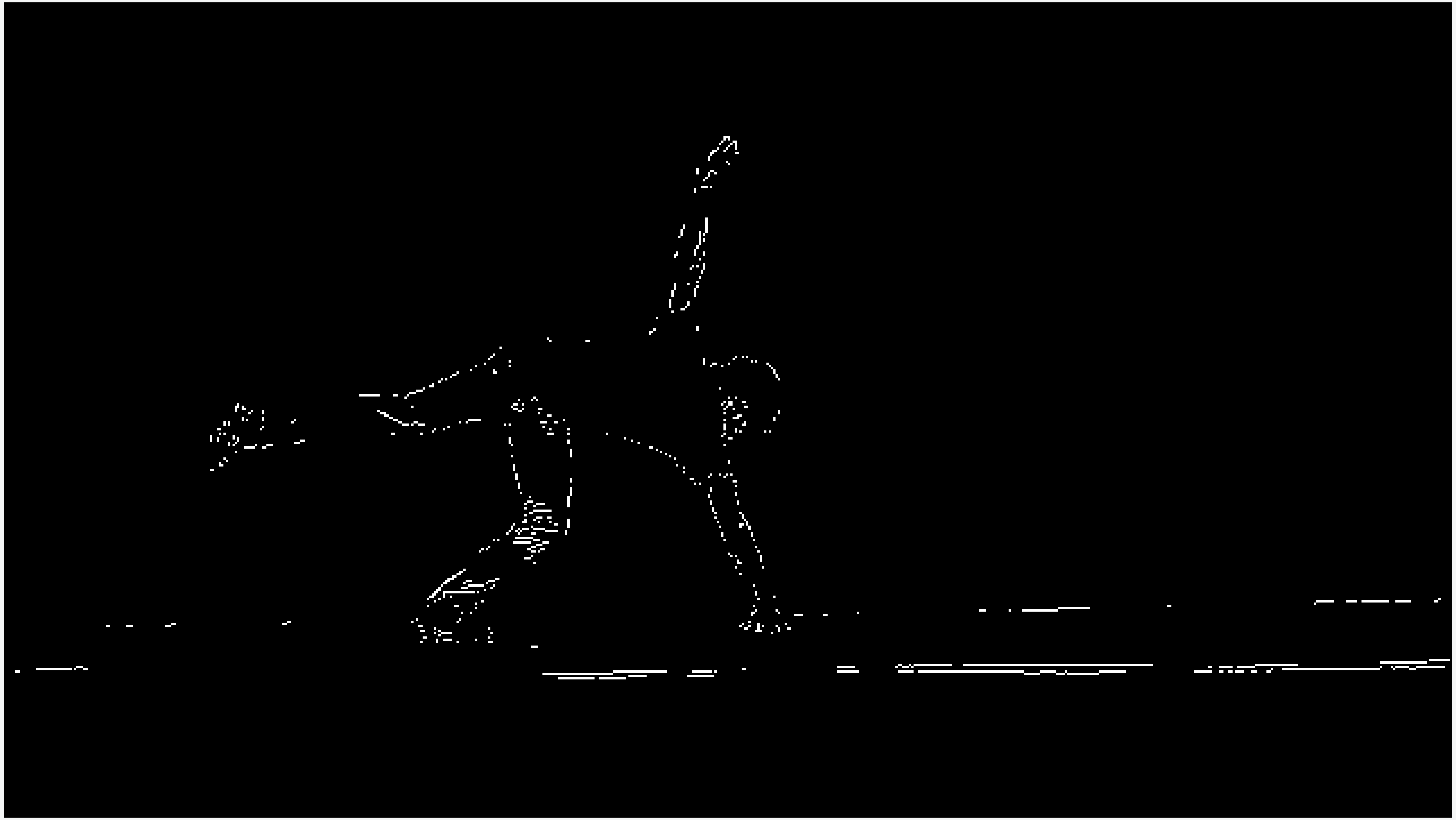


Hysteresis Thresholding

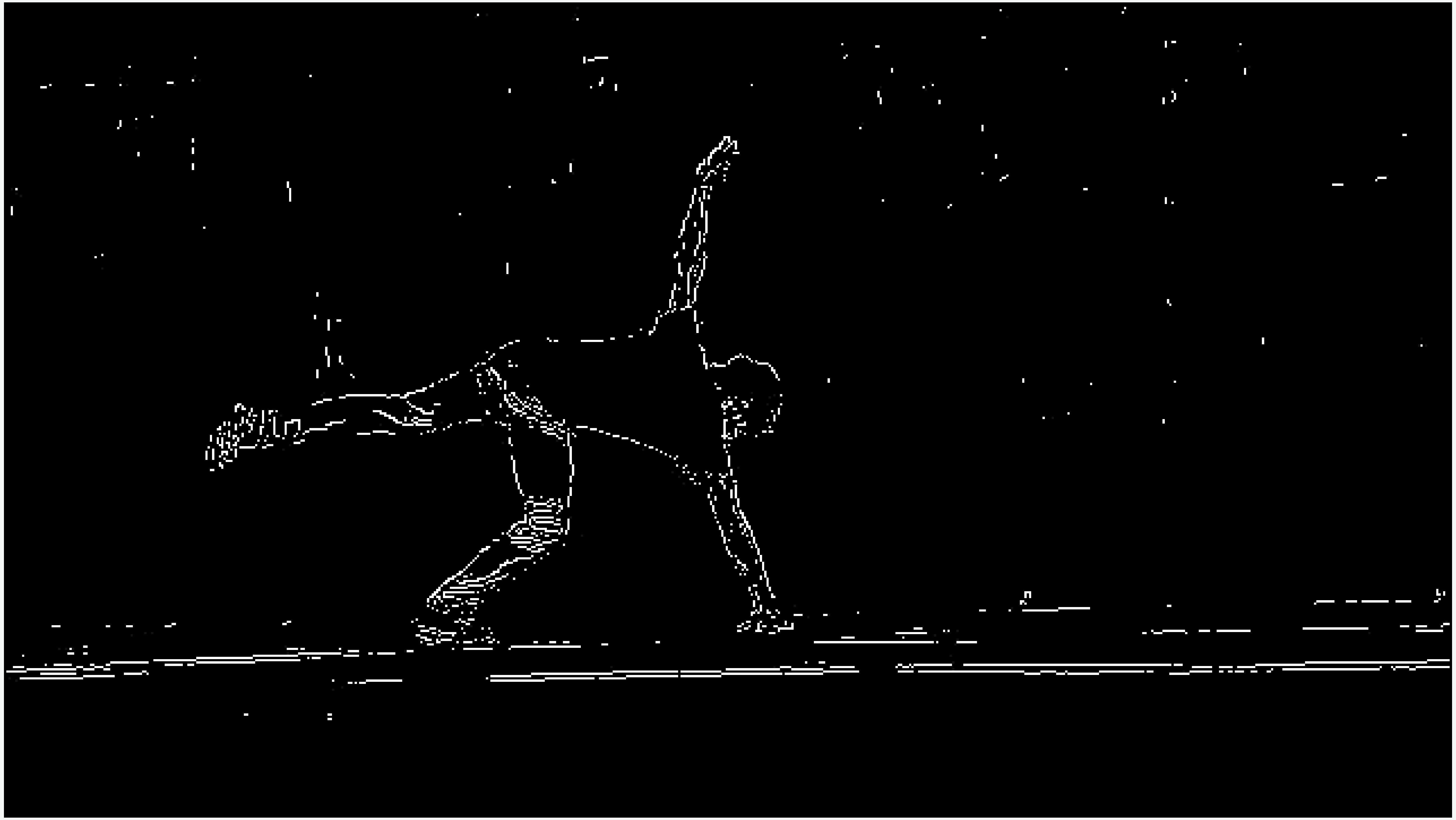


```python
# Result
#sobel_out = cv2.VideoWriter(path+'results/dance_sobel.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
#nms_out = cv2.VideoWriter(path+'results/dance_nms.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
hys_out = cv2.VideoWriter(path+'results/dance_130_80.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)


# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        #--------------------Image preprocessing-------------------------------
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img, sobel_gx)
        #----------------------------------------------------------------------

        #---------------------Image convolution--------------------------------
        sobel_x = convolution(frame, pad_img, sobel_gx)
        sobel_y = convolution(frame, pad_img, sobel_gy)
        sobel, s_theta = magnitude_grad(sobel_x, sobel_y)
        result = sobel.astype(np.uint8)
        #sobel_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Non Maximum Suppression--------------------------
        nms_img = grad_nms(sobel, s_theta)
        result = nms_img.astype(np.uint8)
        #nms_out.write(result)
        #----------------------------------------------------------------------

        #---------------------Hysteresis Thresholding--------------------------
        double = threshold(result, 130, 80)
        result = hysteresis(double)
        result = result.astype(np.uint8)
        hys_out.write(result)
        #----------------------------------------------------------------------

    else:
        break

#sobel_out.release()
#nms_out.release()
hys_out.release()
print('done')
```

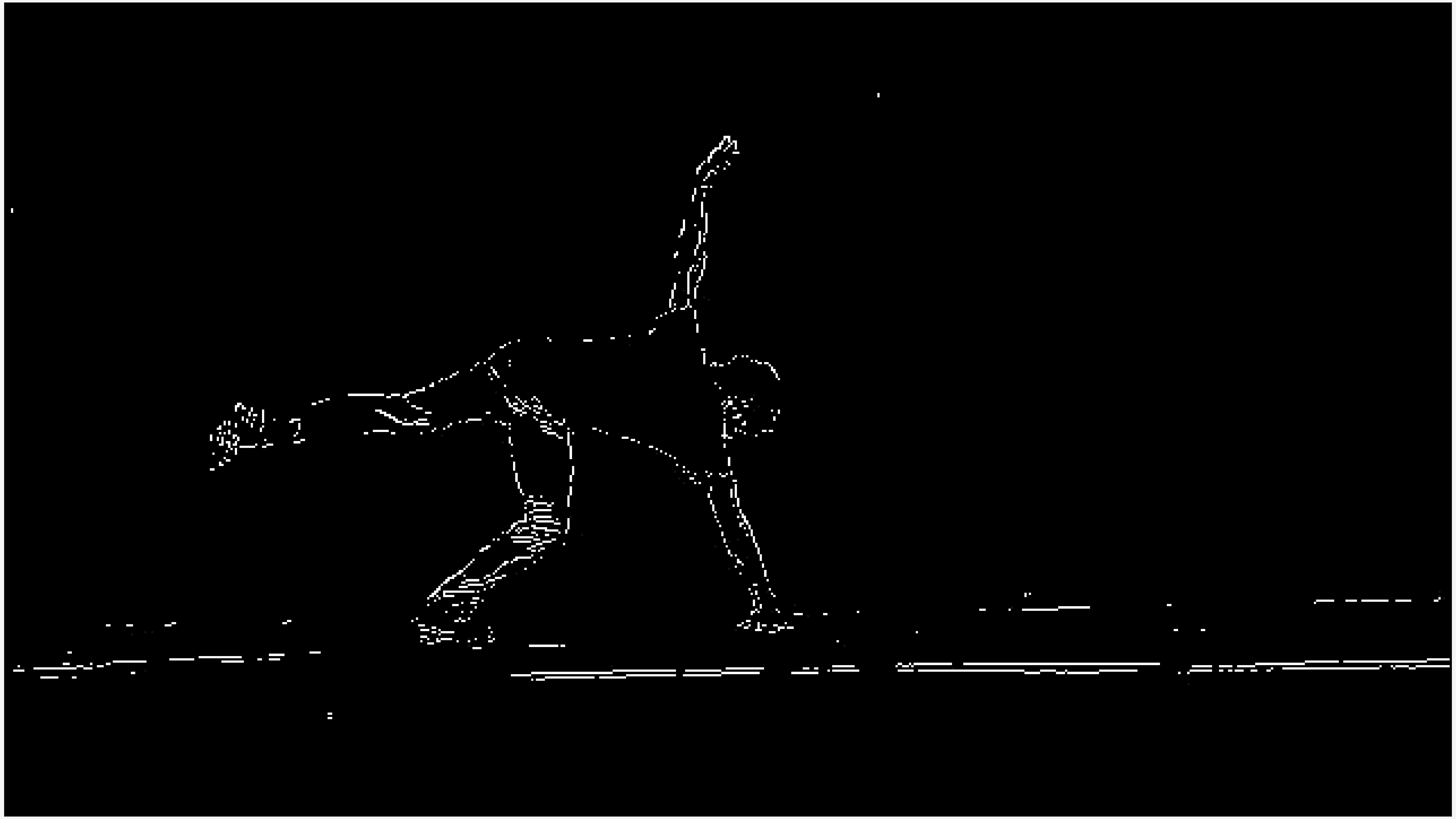
High : 170, Low : 80



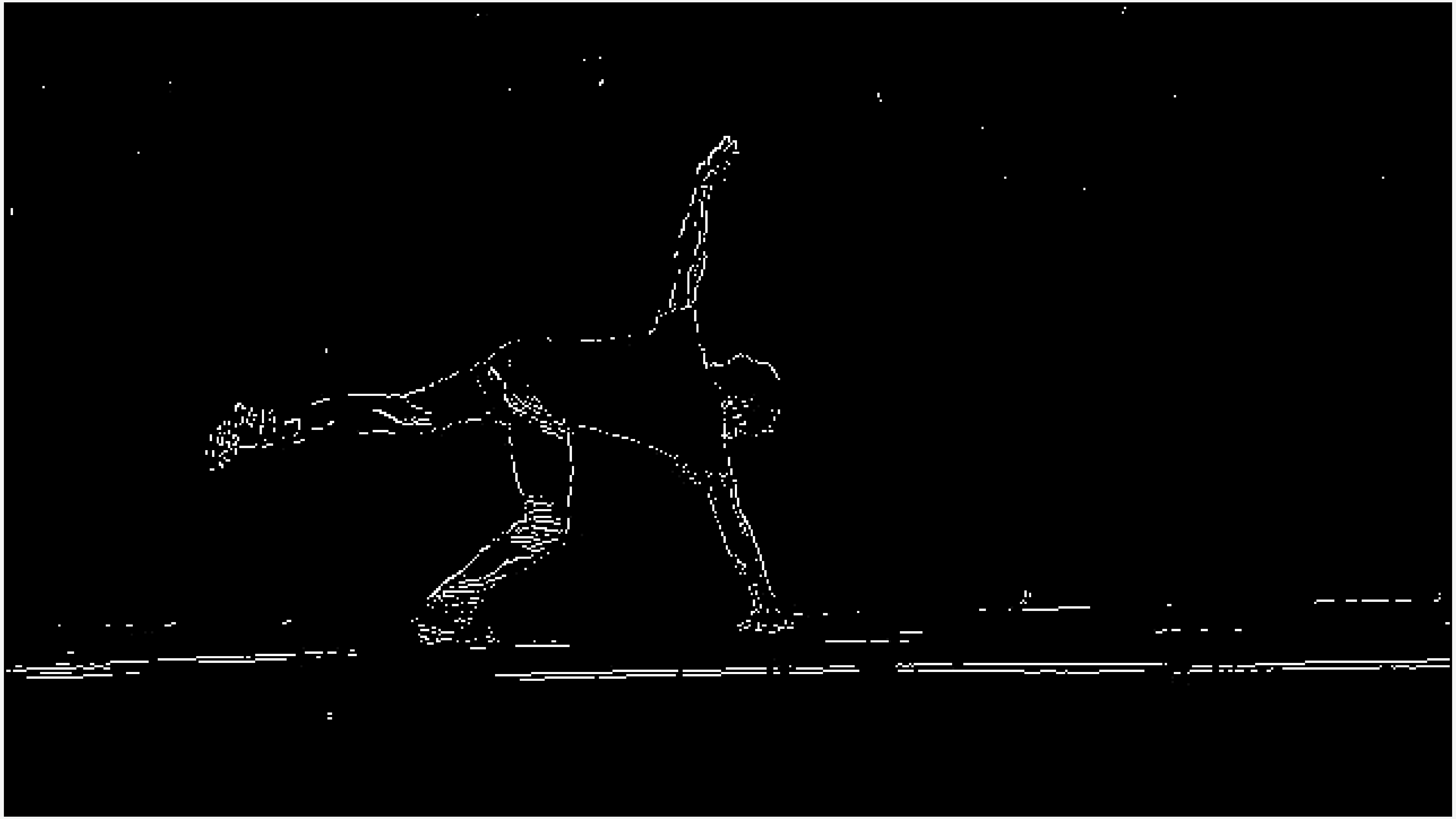
High : 90, Low : 80



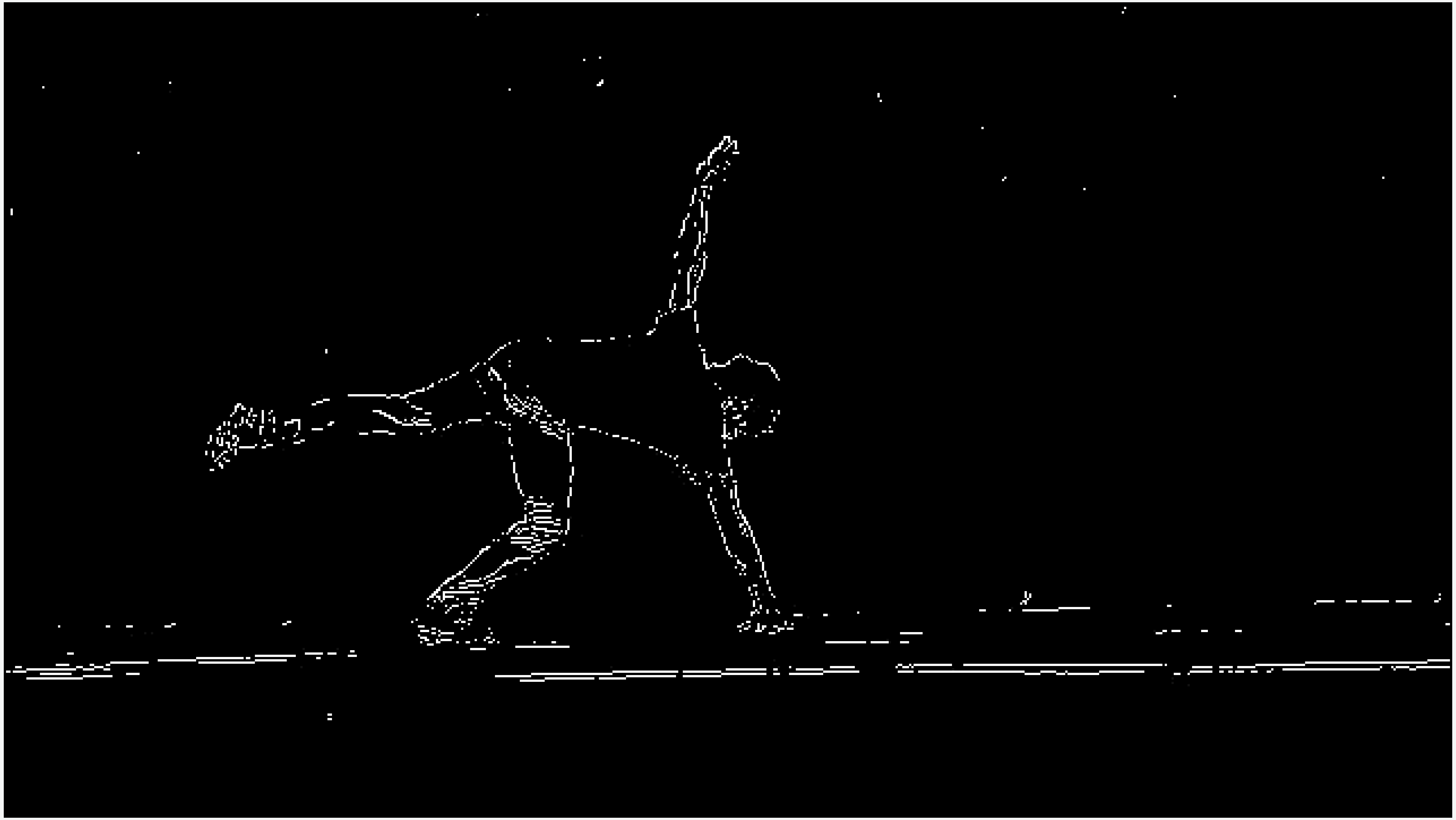
High : 130, Low : 80



High : 110, Low : 80

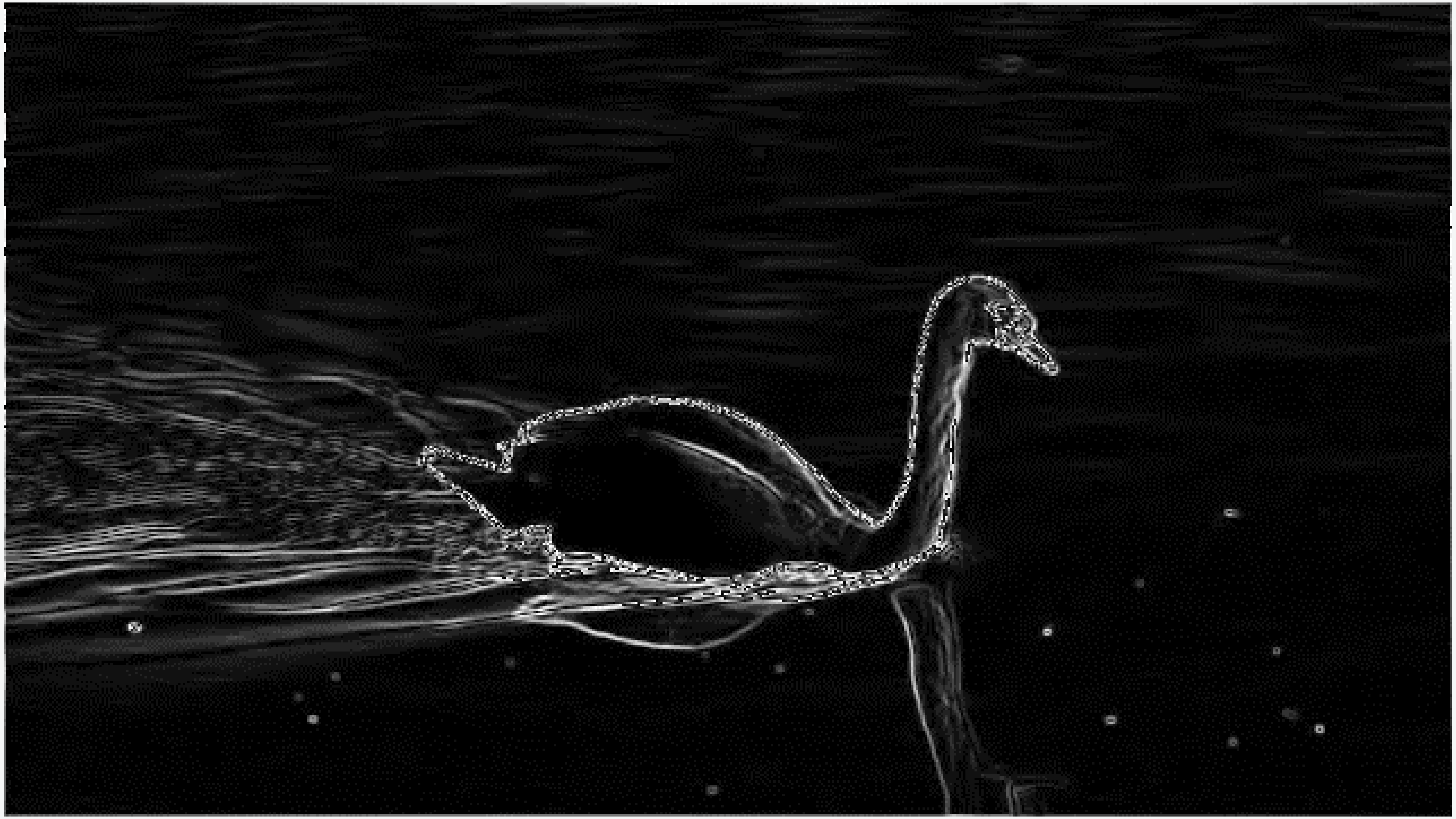


High : 110, Low : 50

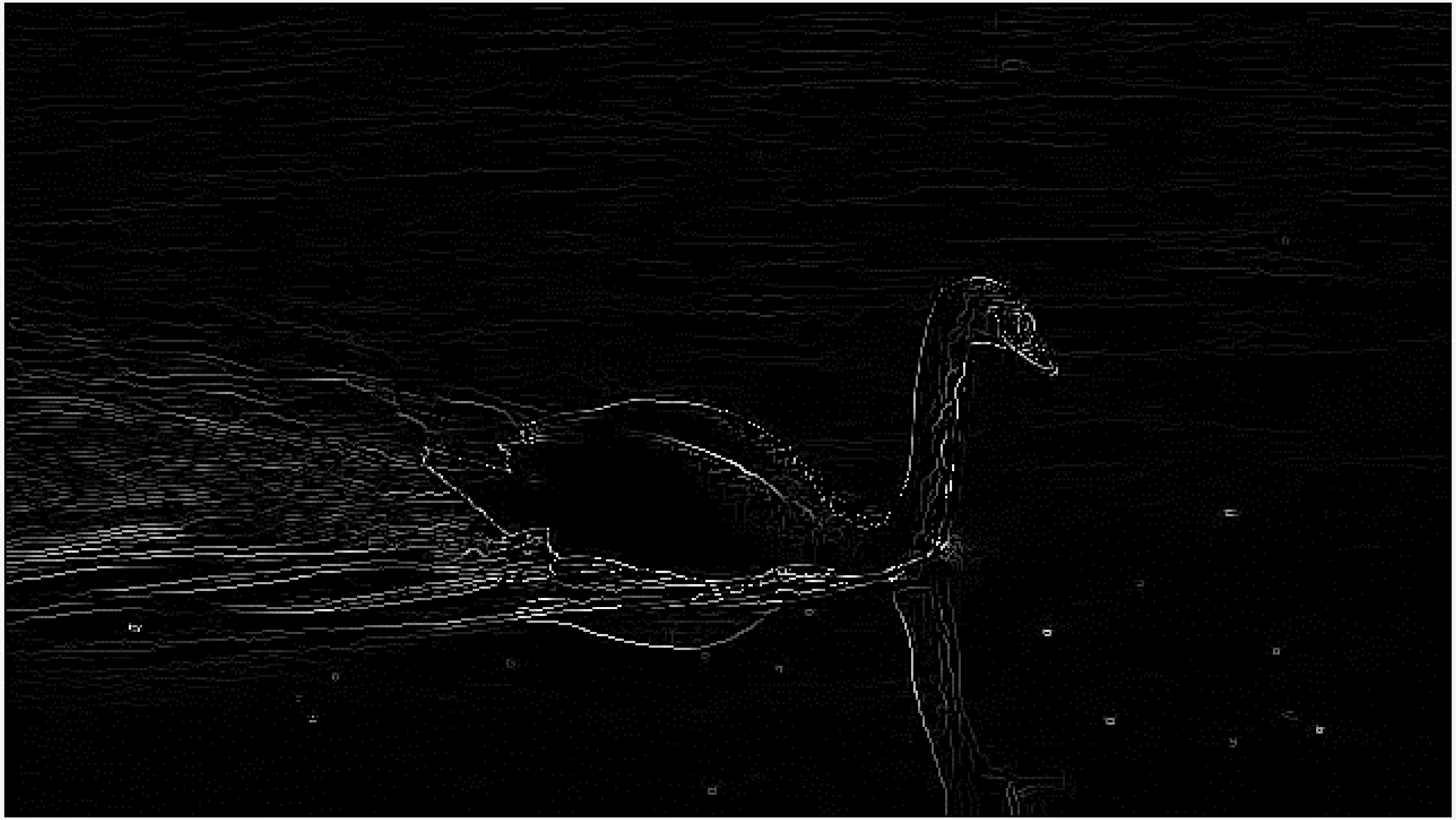


2) Second Video

Sobel Mask

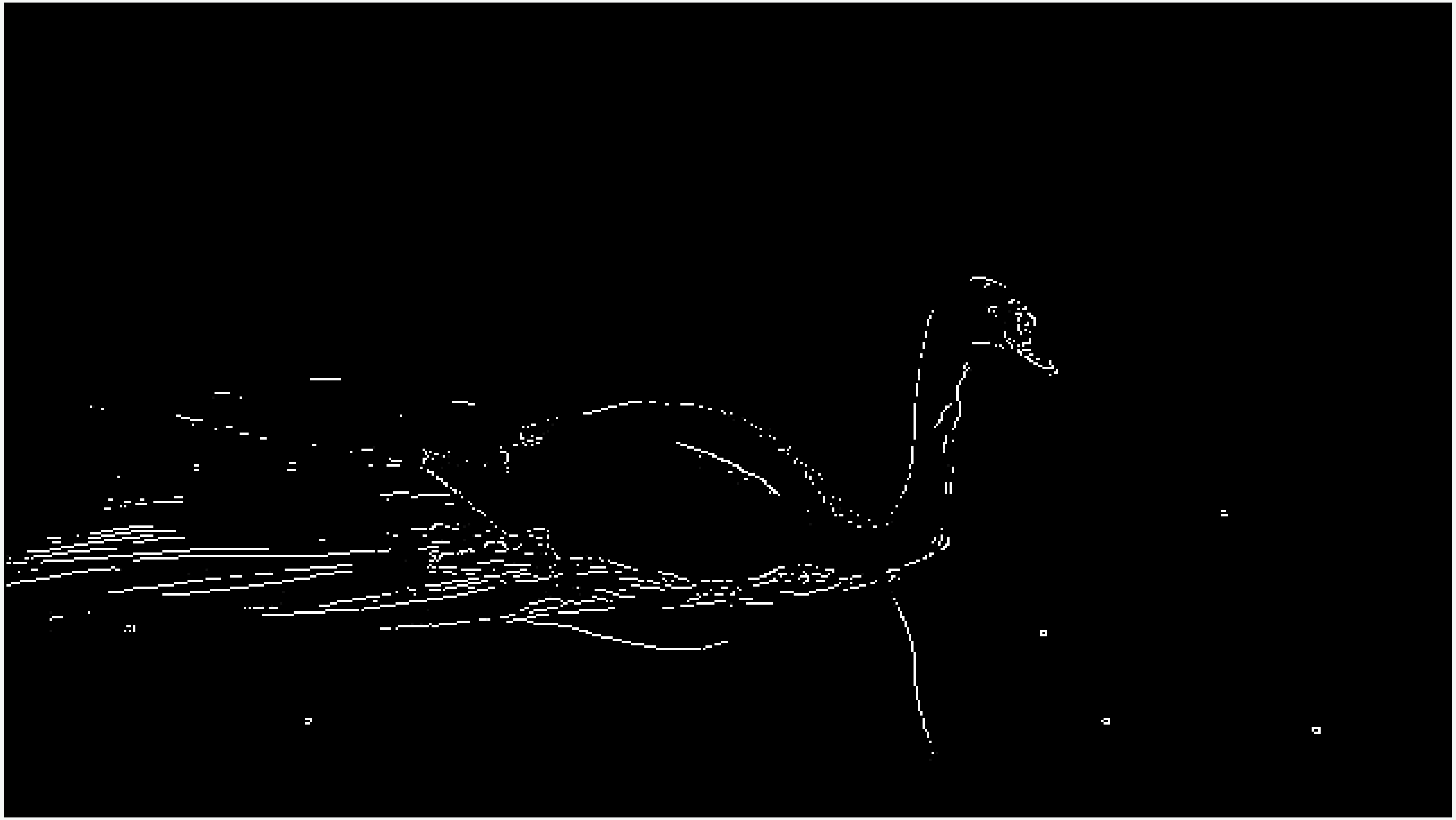


Non Maximum Suppression



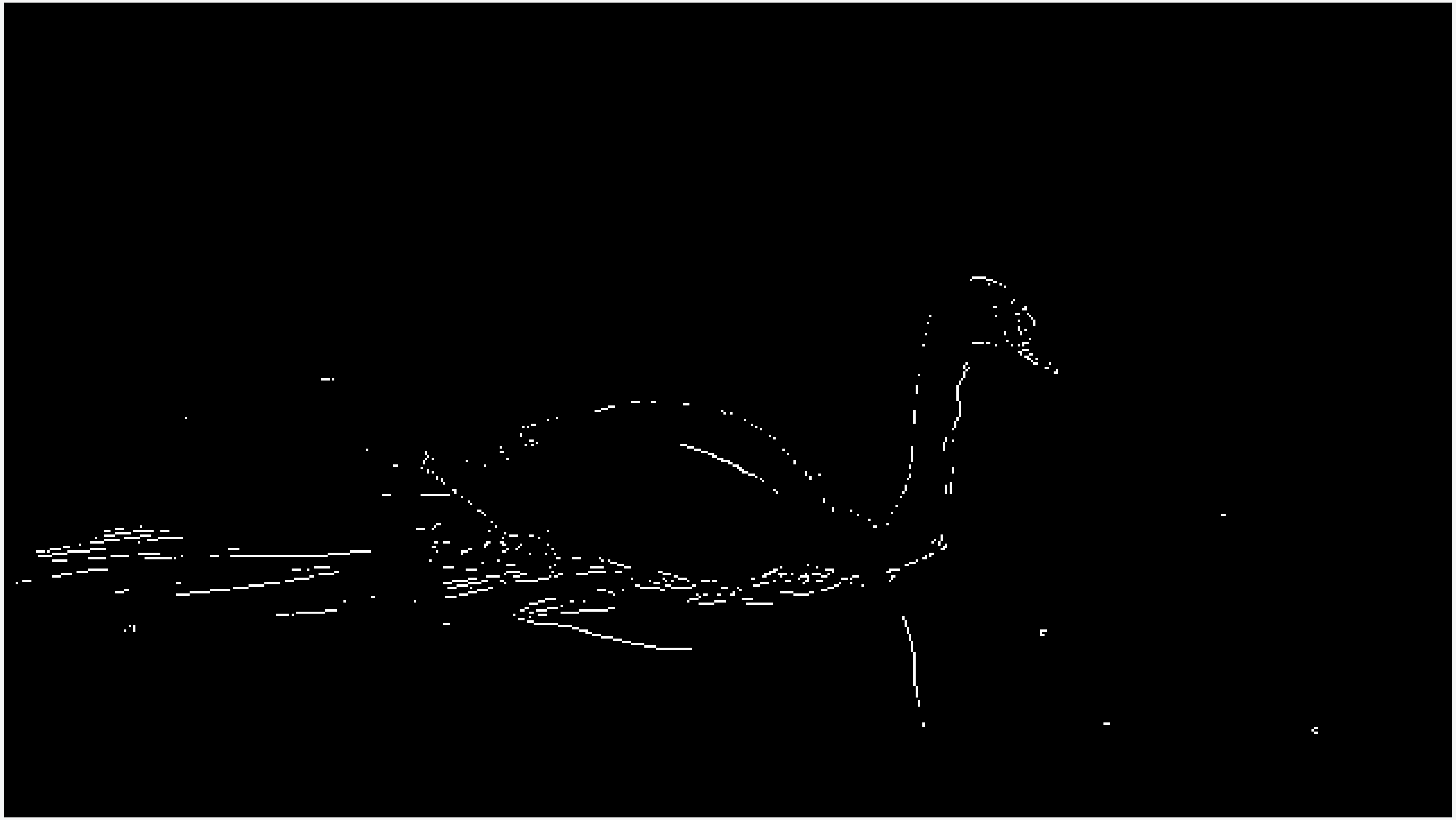
Hysteresis Thresholding

High : 110, Low : 50



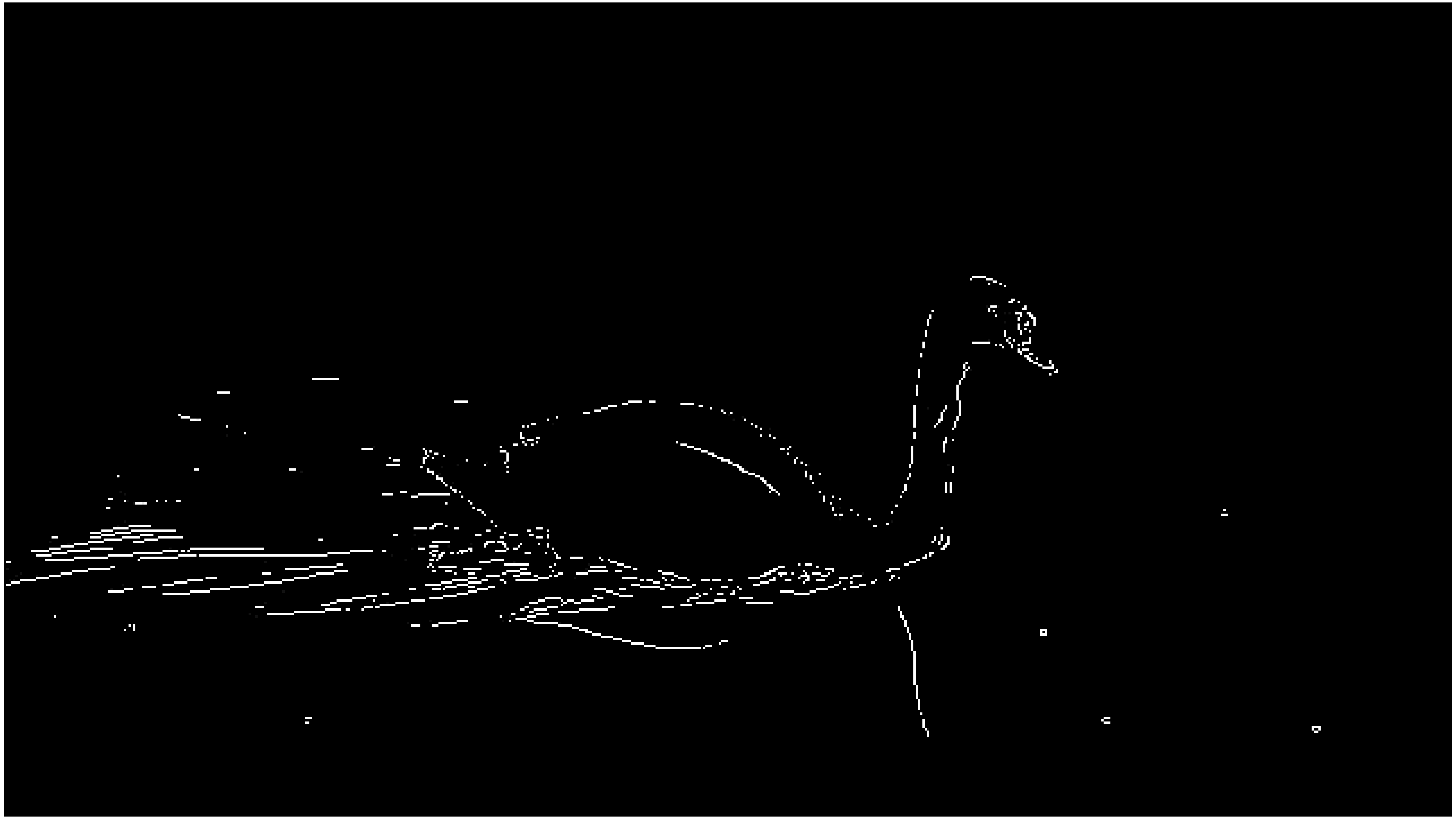
Hysteresis Thresholding

High : 150, Low : 50



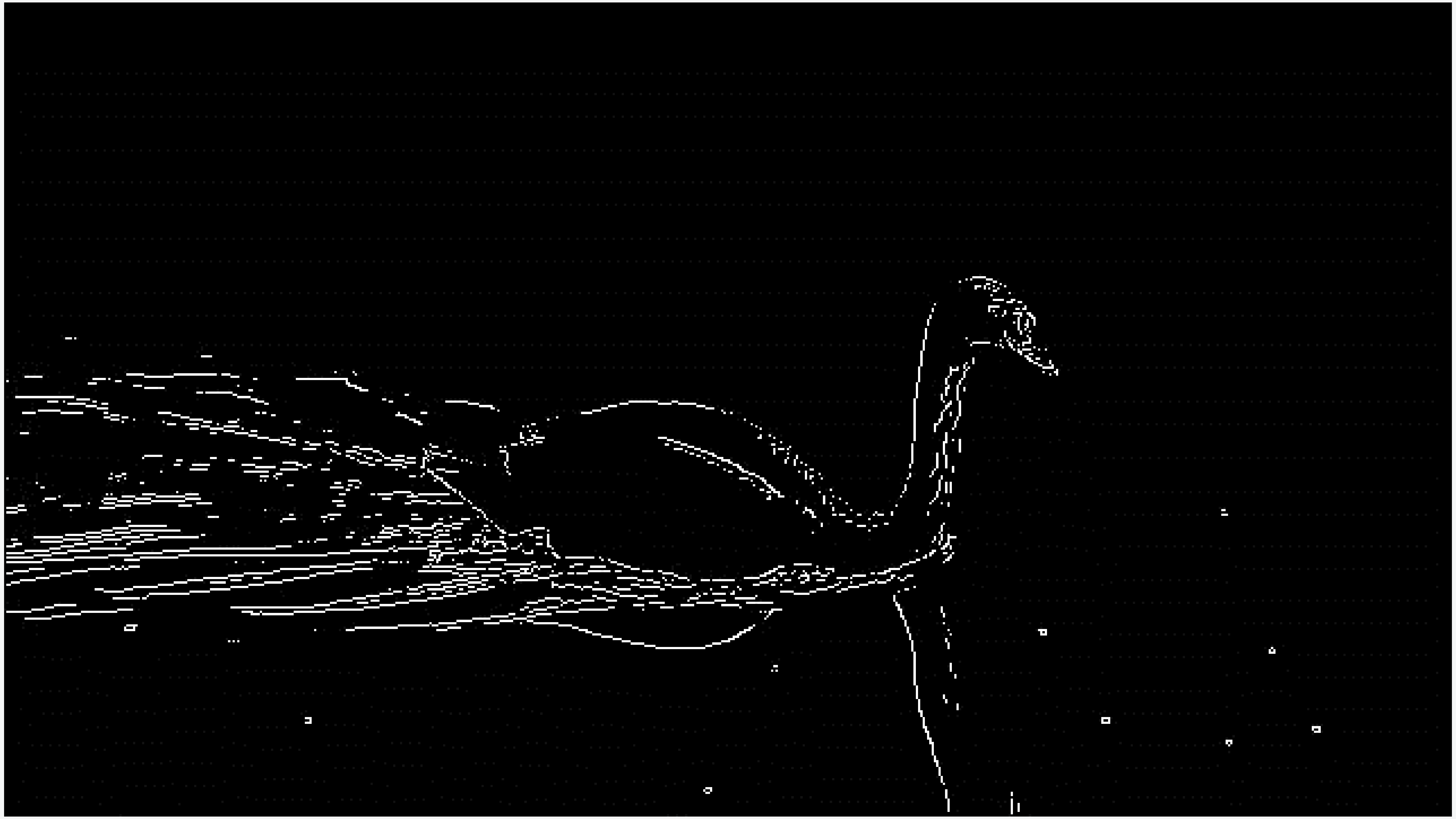
Hysteresis Thresholding

High : 120, Low : 50



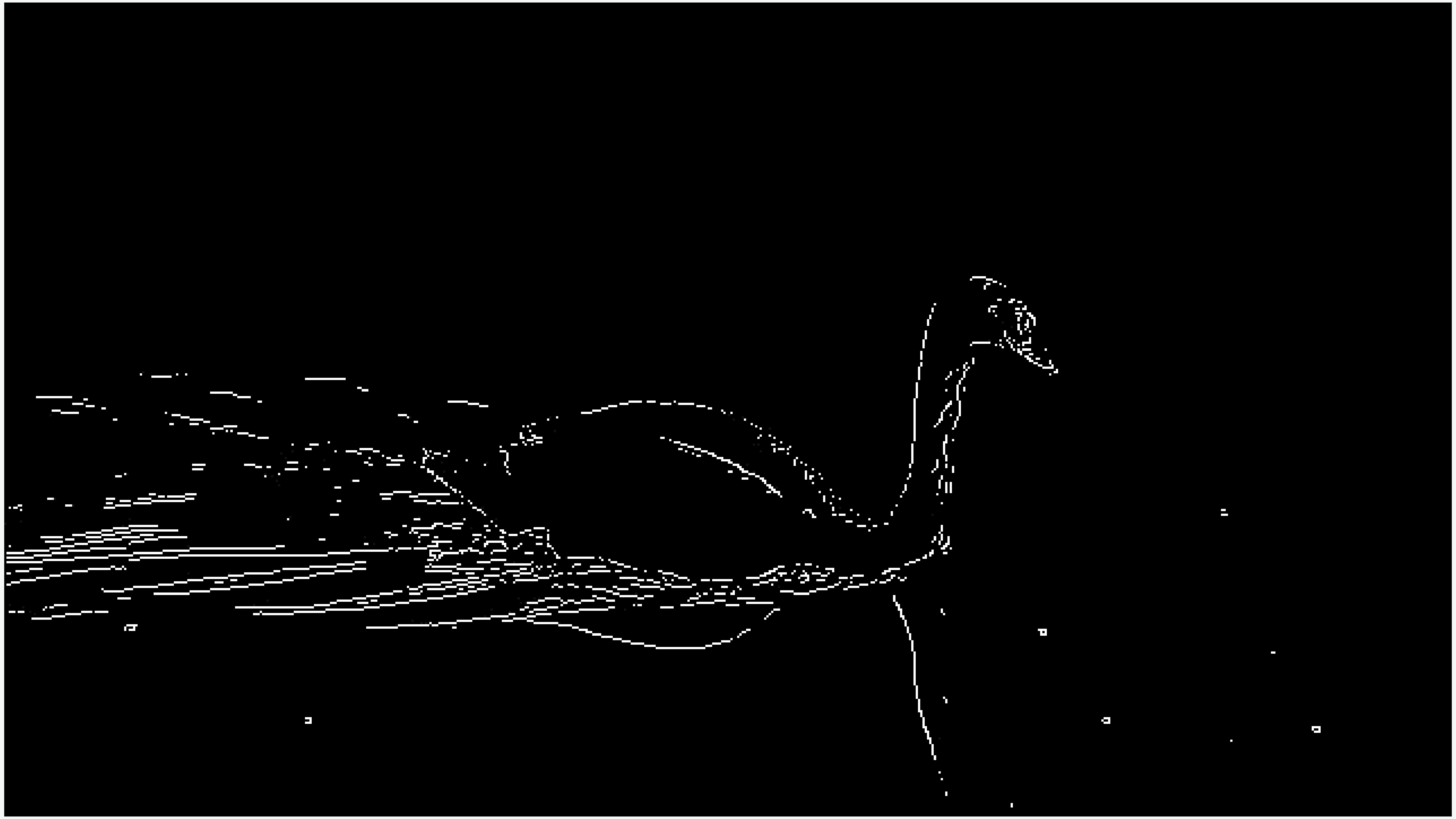
Hysteresis Thresholding

High : 70, Low : 30



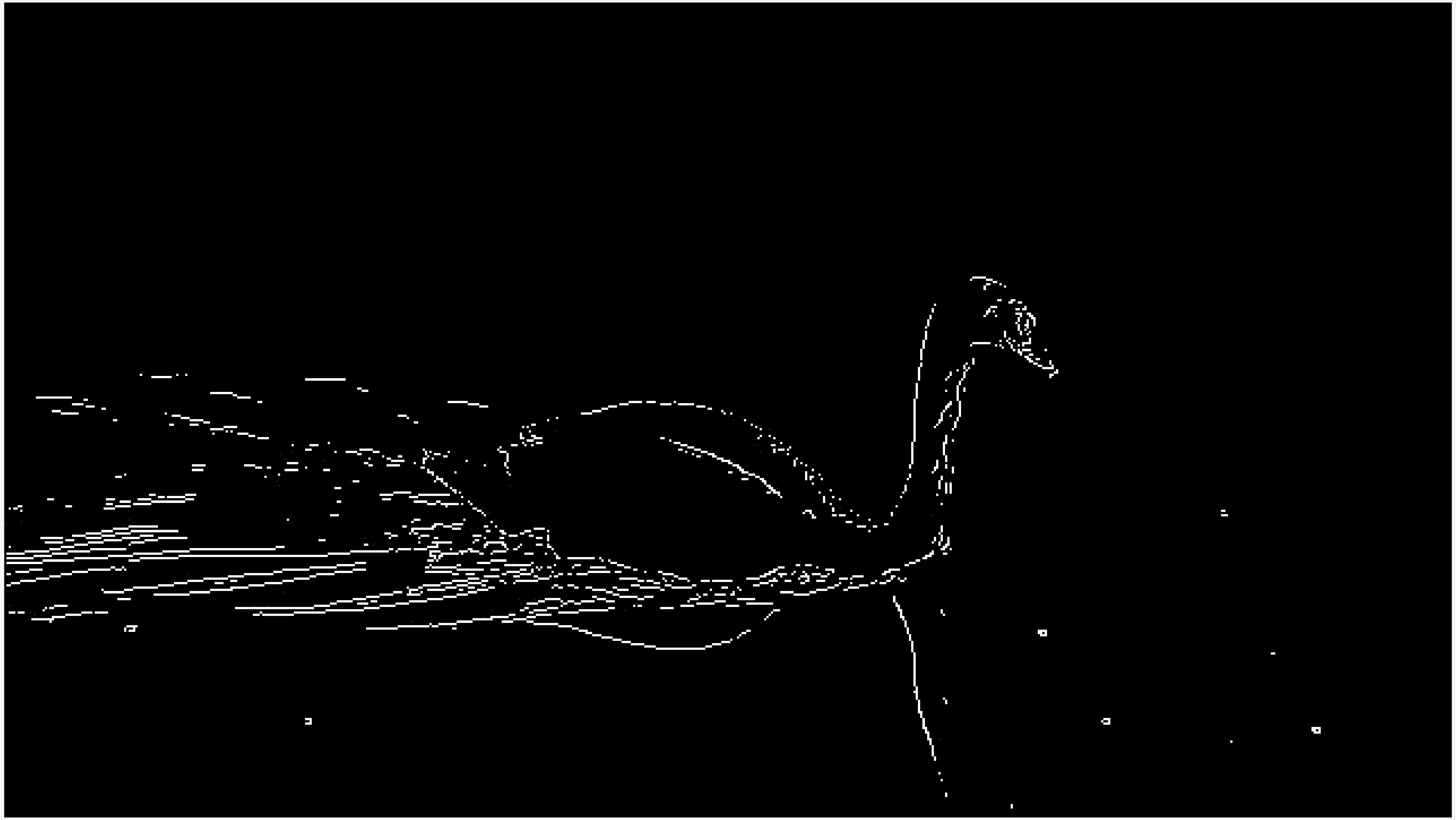
Hysteresis Thresholding

High : 90, Low : 30



Hysteresis Thresholding

High : 90, Low : 10

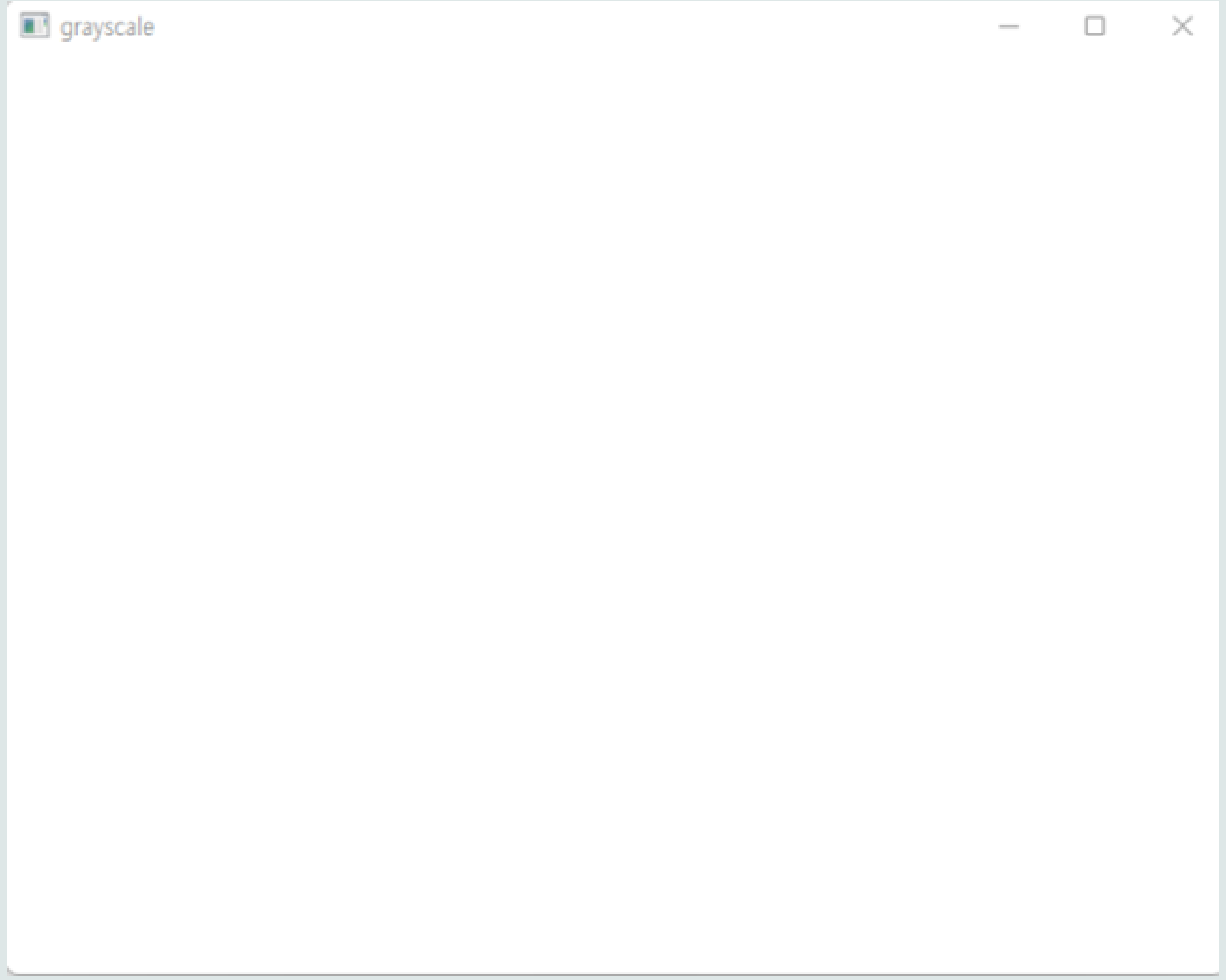


07. What I tried

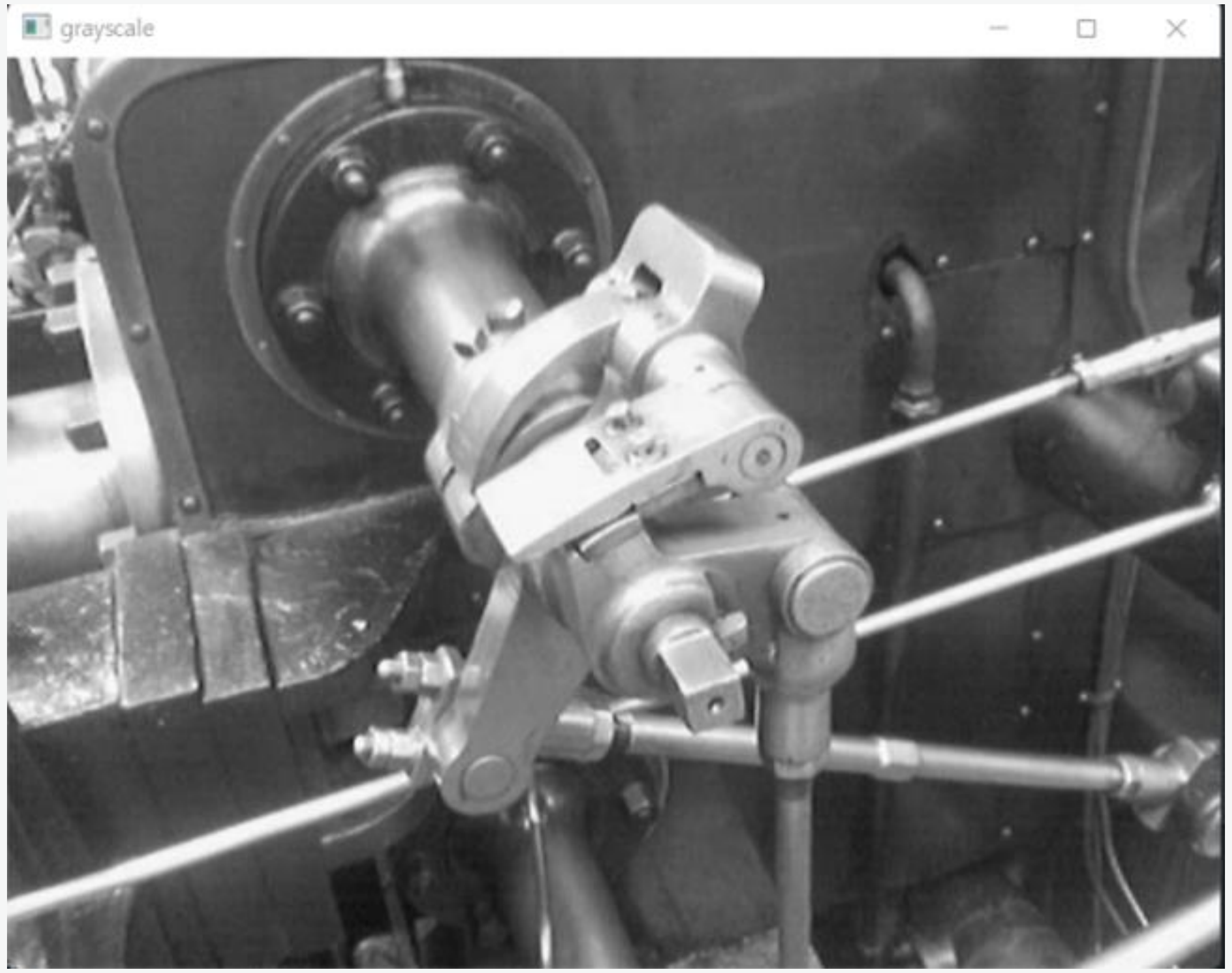
1. Normalization Problem

2. Video Processing

1) Normalization Problem



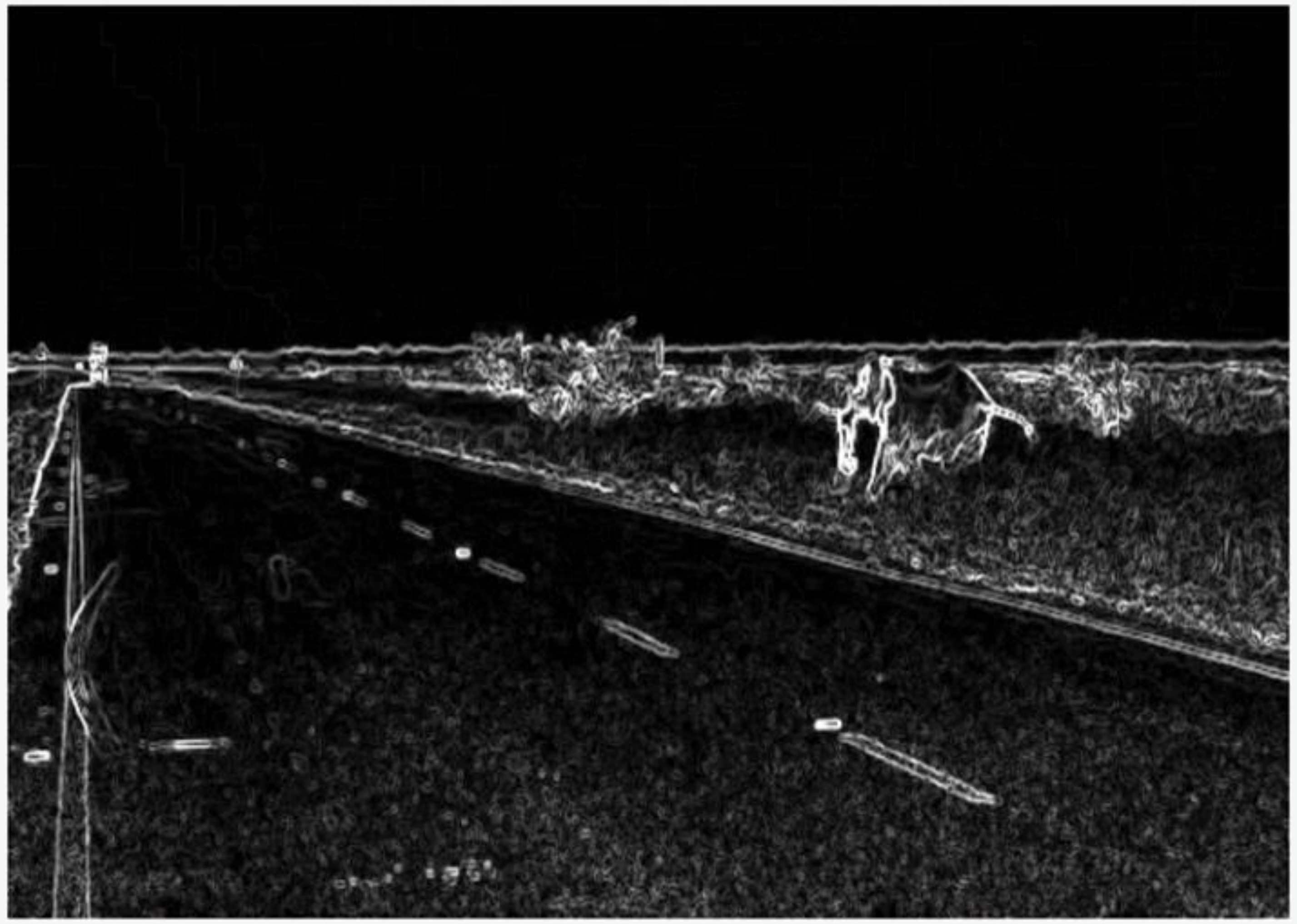
Without normalization



With normalization

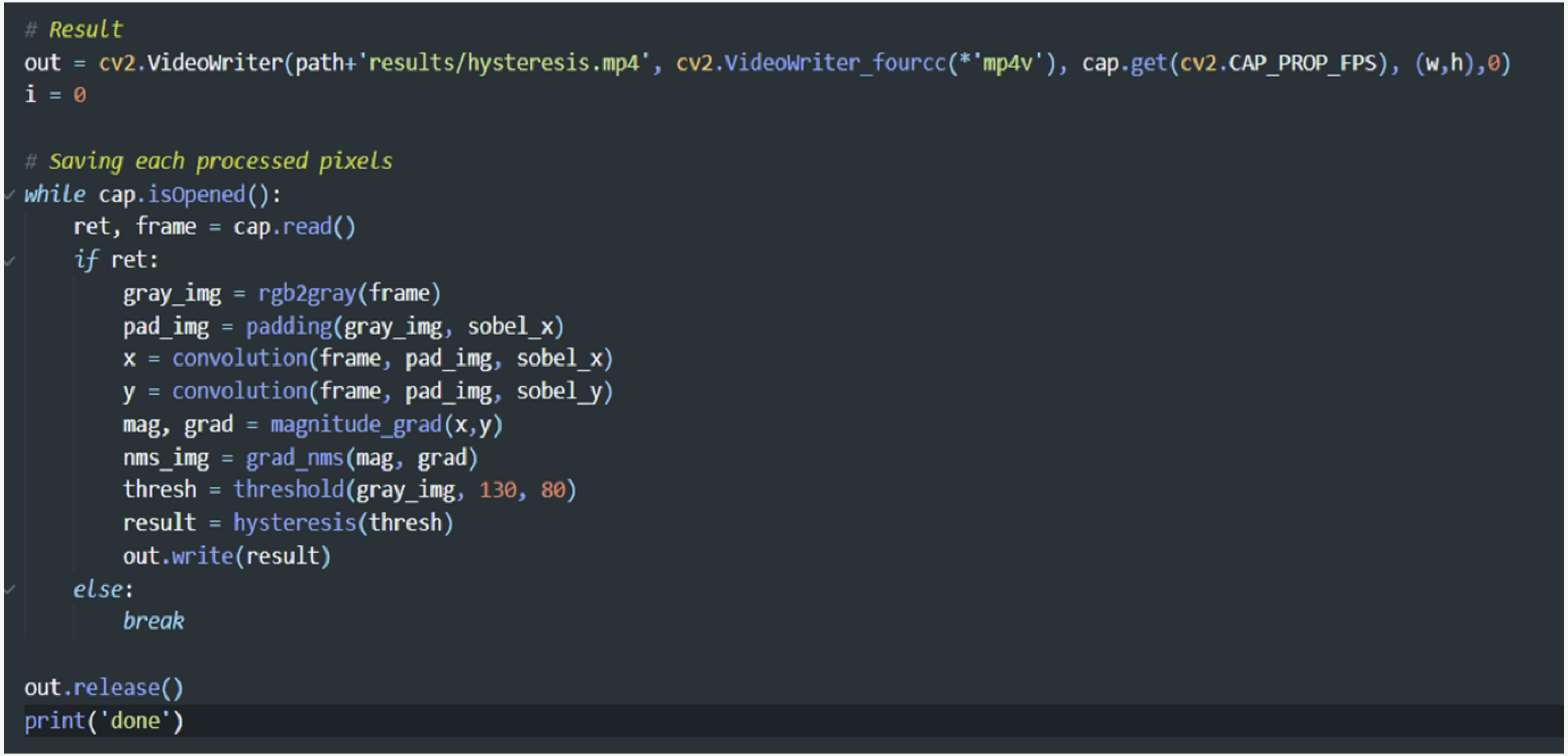
1) Normalization Problem

Without normalization

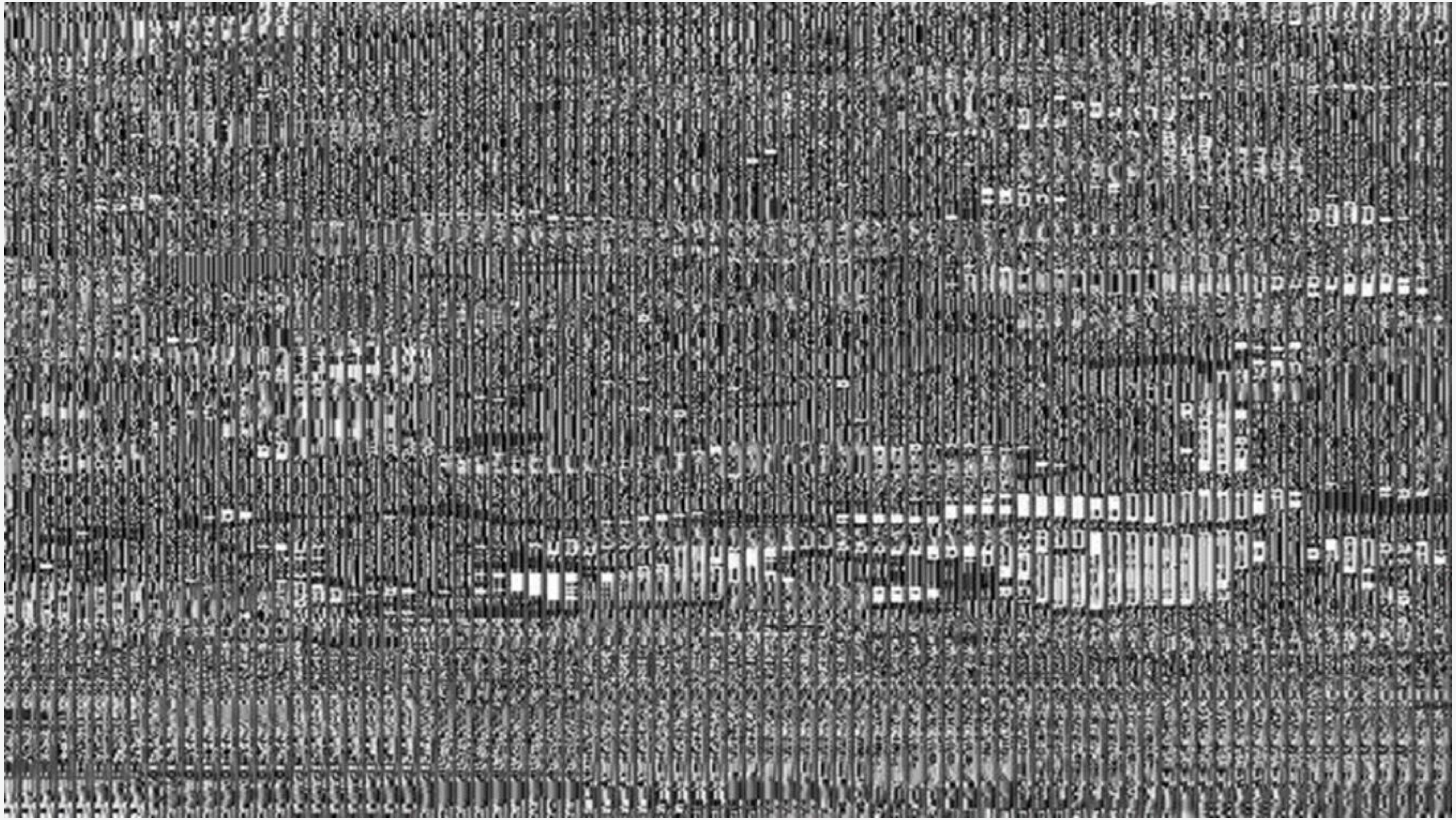


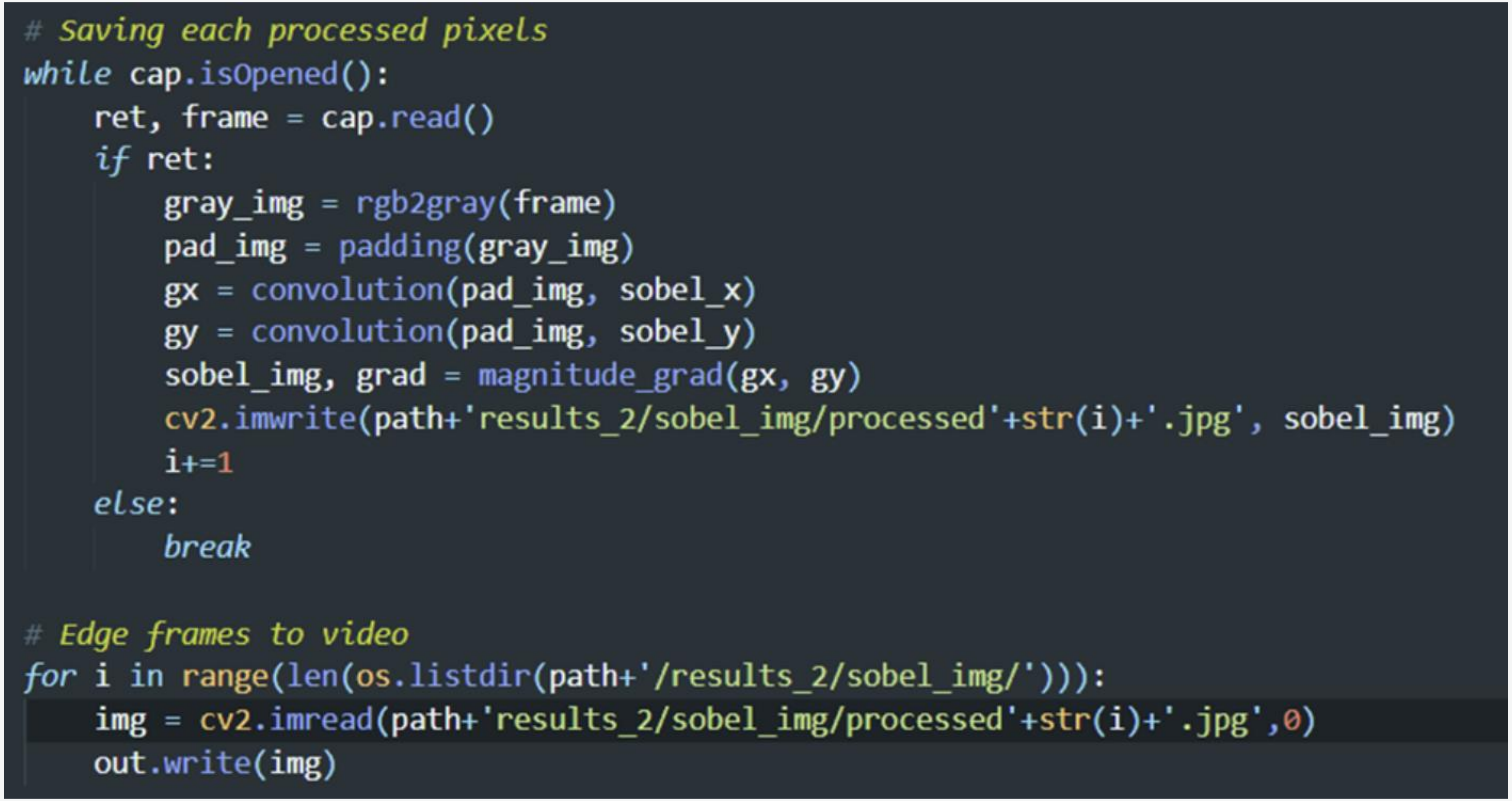
With normalization

2) Video Processing



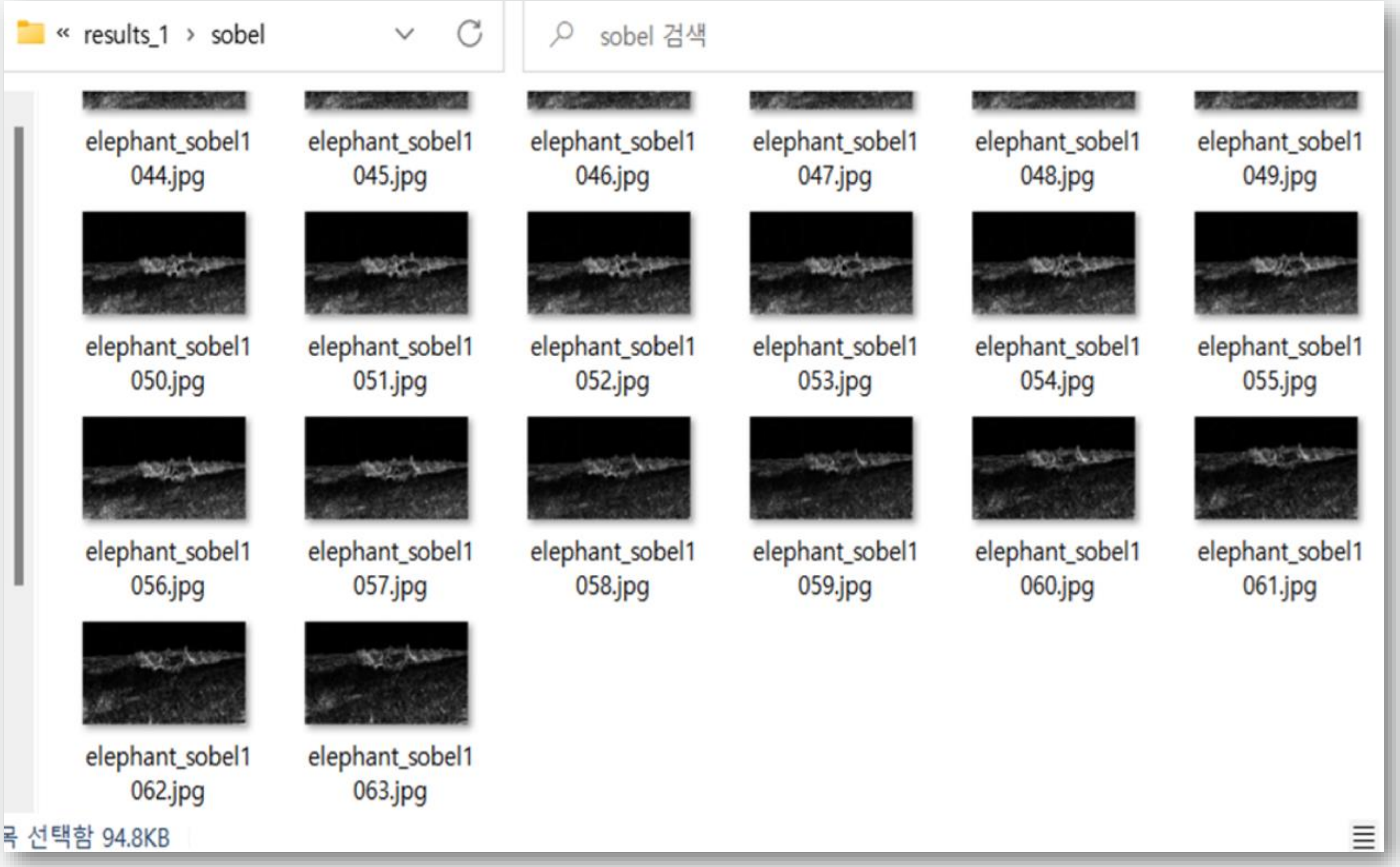
```python
# Result
out = cv2.VideoWriter(path+'results/hysteresis.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
i = 0

# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img, sobel_x)
        x = convolution(frame, pad_img, sobel_x)
        y = convolution(frame, pad_img, sobel_y)
        mag, grad = magnitude_grad(x,y)
        nms_img = grad_nms(mag, grad)
        thresh = threshold(gray_img, 130, 80)
        result = hysteresis(thresh)
        out.write(result)
    else:
        break

out.release()
print('done')
```





```
# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img)
        gx = convolution(pad_img, sobel_x)
        gy = convolution(pad_img, sobel_y)
        sobel_img, grad = magnitude_grad(gx, gy)
        cv2.imwrite(path+'results_2/sobel_img/processed'+str(i)+'.jpg', sobel_img)
        i+=1
    else:
        break

# Edge frames to video
for i in range(len(os.listdir(path+'/results_2/sobel_img/'))):
    img = cv2.imread(path+'results_2/sobel_img/processed'+str(i)+'.jpg',0)
    out.write(img)
```

07. What I tried

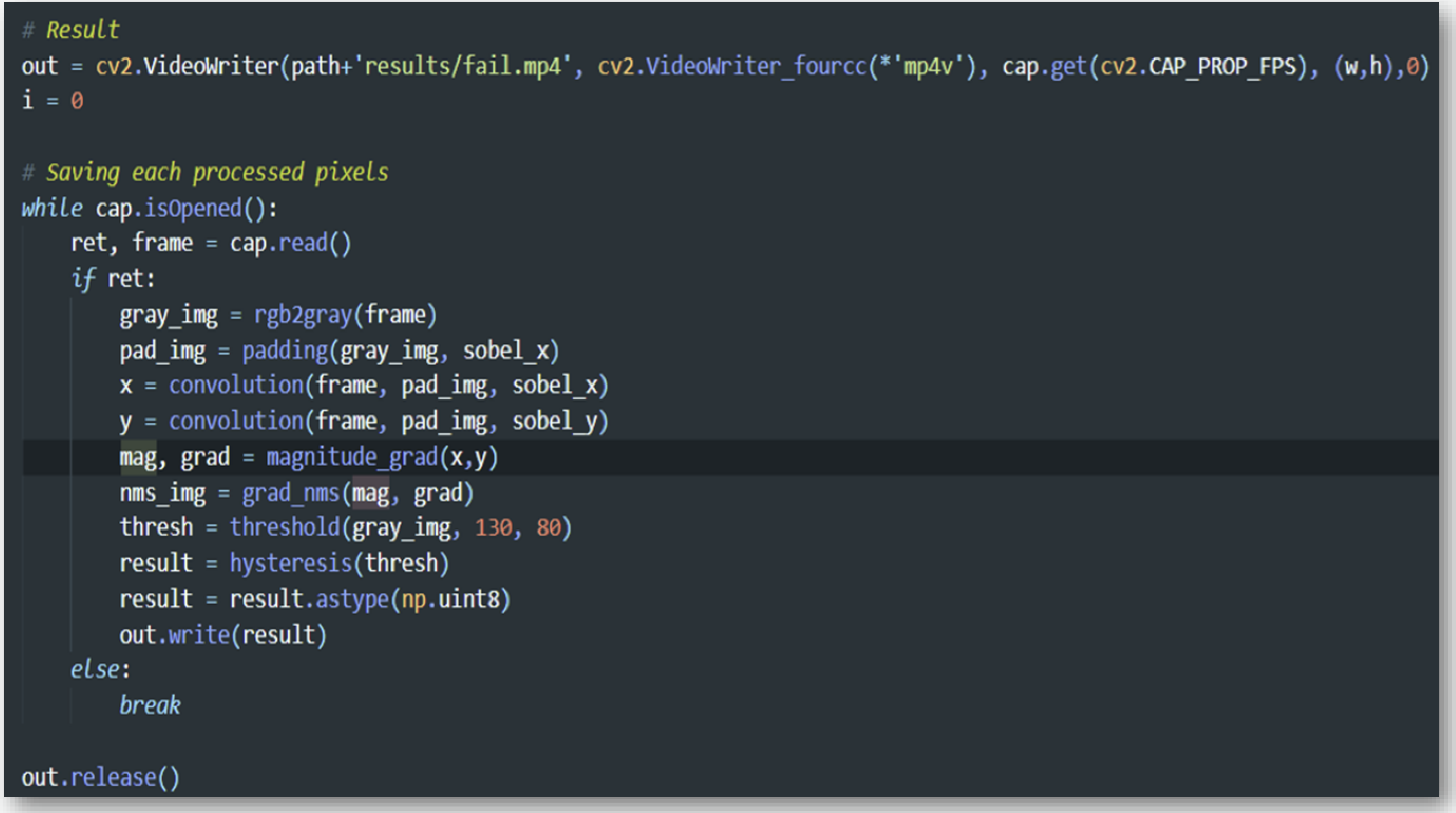


« results_1 › sobel

elephant_sobel1044.jpg
elephant_sobel1045.jpg
elephant_sobel1046.jpg
elephant_sobel1047.jpg
elephant_sobel1048.jpg
elephant_sobel1049.jpg
elephant_sobel1050.jpg
elephant_sobel1051.jpg
elephant_sobel1052.jpg
elephant_sobel1053.jpg
elephant_sobel1054.jpg
elephant_sobel1055.jpg
elephant_sobel1056.jpg
elephant_sobel1057.jpg
elephant_sobel1058.jpg
elephant_sobel1059.jpg
elephant_sobel1060.jpg
elephant_sobel1061.jpg
elephant_sobel1062.jpg
elephant_sobel1063.jpg

목 선택함 94.8KB

07. What I tried



```
# Result
out = cv2.VideoWriter(path+'results/fail.mp4', cv2.VideoWriter_fourcc(*'mp4v'), cap.get(cv2.CAP_PROP_FPS), (w,h),0)
i = 0

# Saving each processed pixels
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        gray_img = rgb2gray(frame)
        pad_img = padding(gray_img, sobel_x)
        x = convolution(frame, pad_img, sobel_x)
        y = convolution(frame, pad_img, sobel_y)
        mag, grad = magnitude_grad(x,y)
        nms_img = grad_nms(mag, grad)
        thresh = threshold(gray_img, 130, 80)
        result = hysteresis(thresh)
        result = result.astype(np.uint8)
        out.write(result)
    else:
        break

out.release()
```

감사합니다.