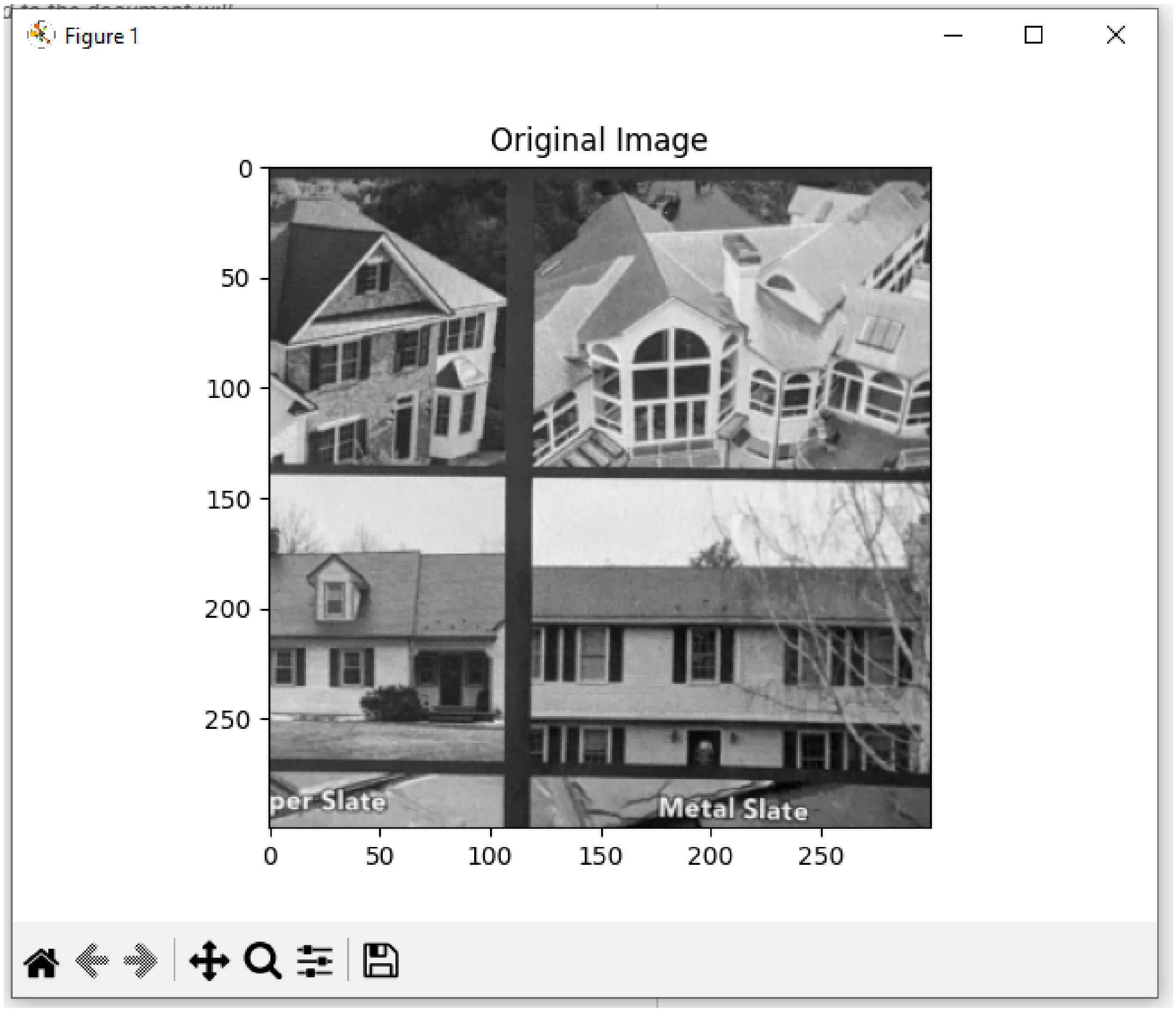
William Dang 113448207



Original Image, pic1grey300.jpg

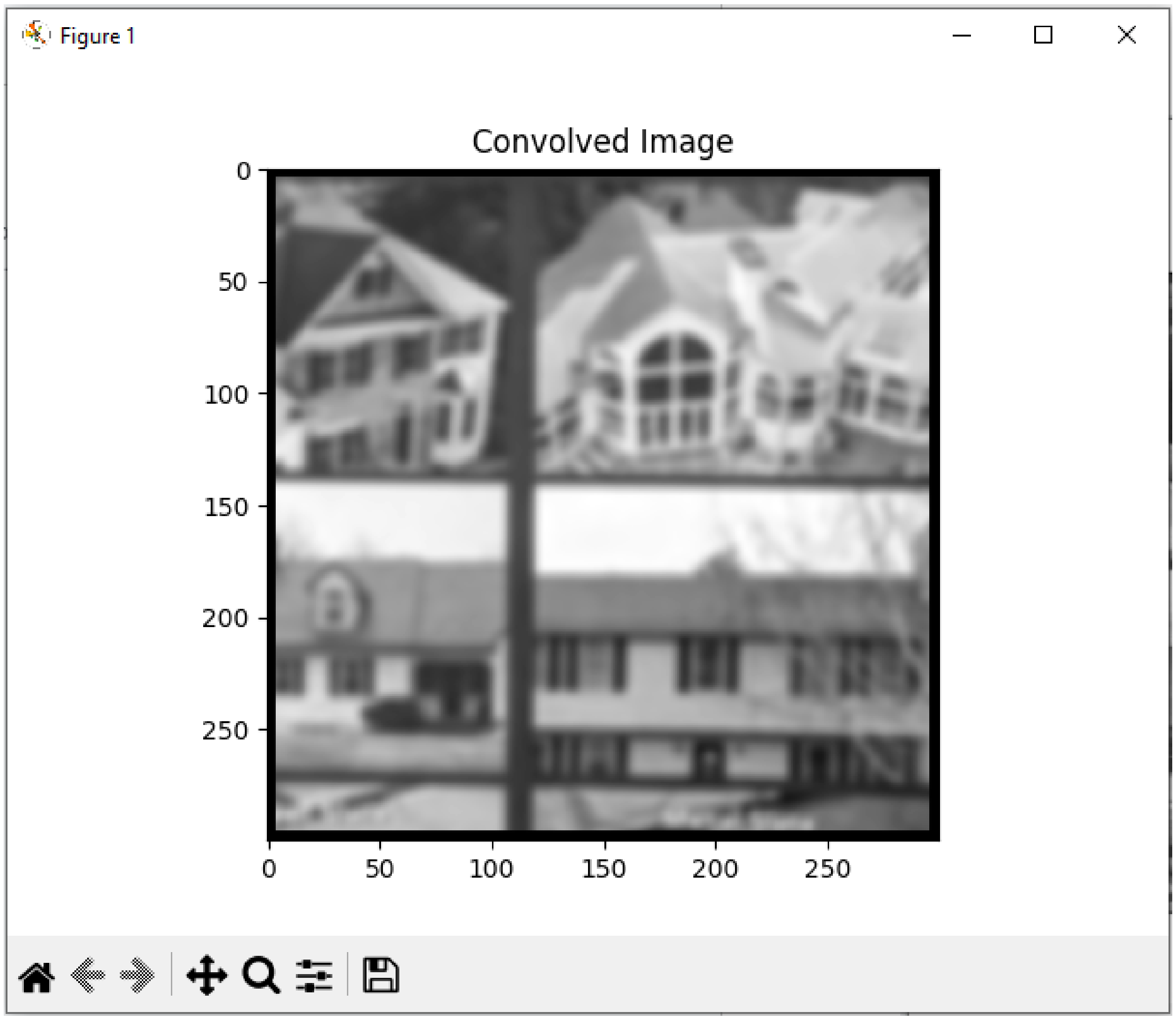


Image of Convolved Image for Part1

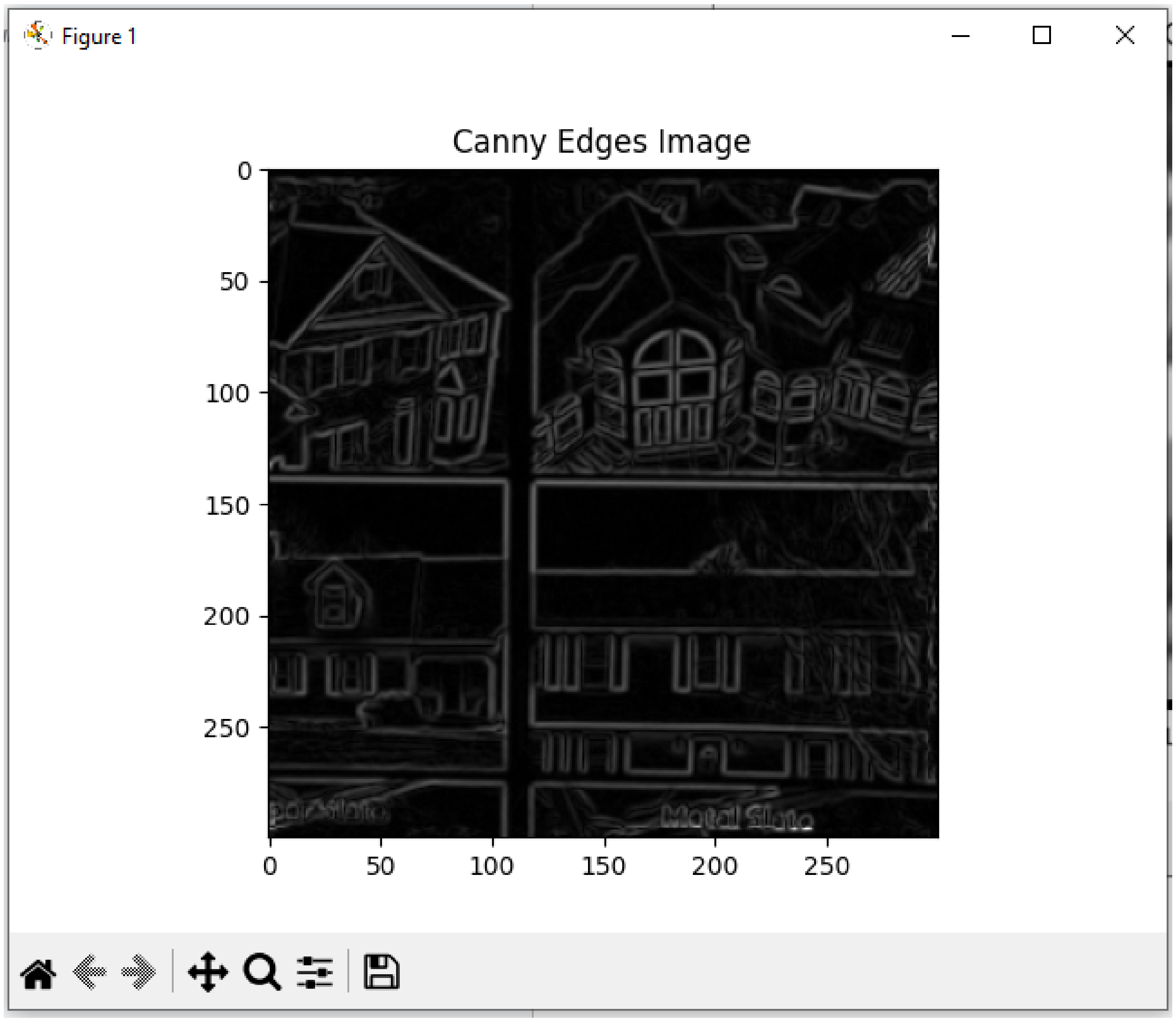


Image of Canny edges for Part2

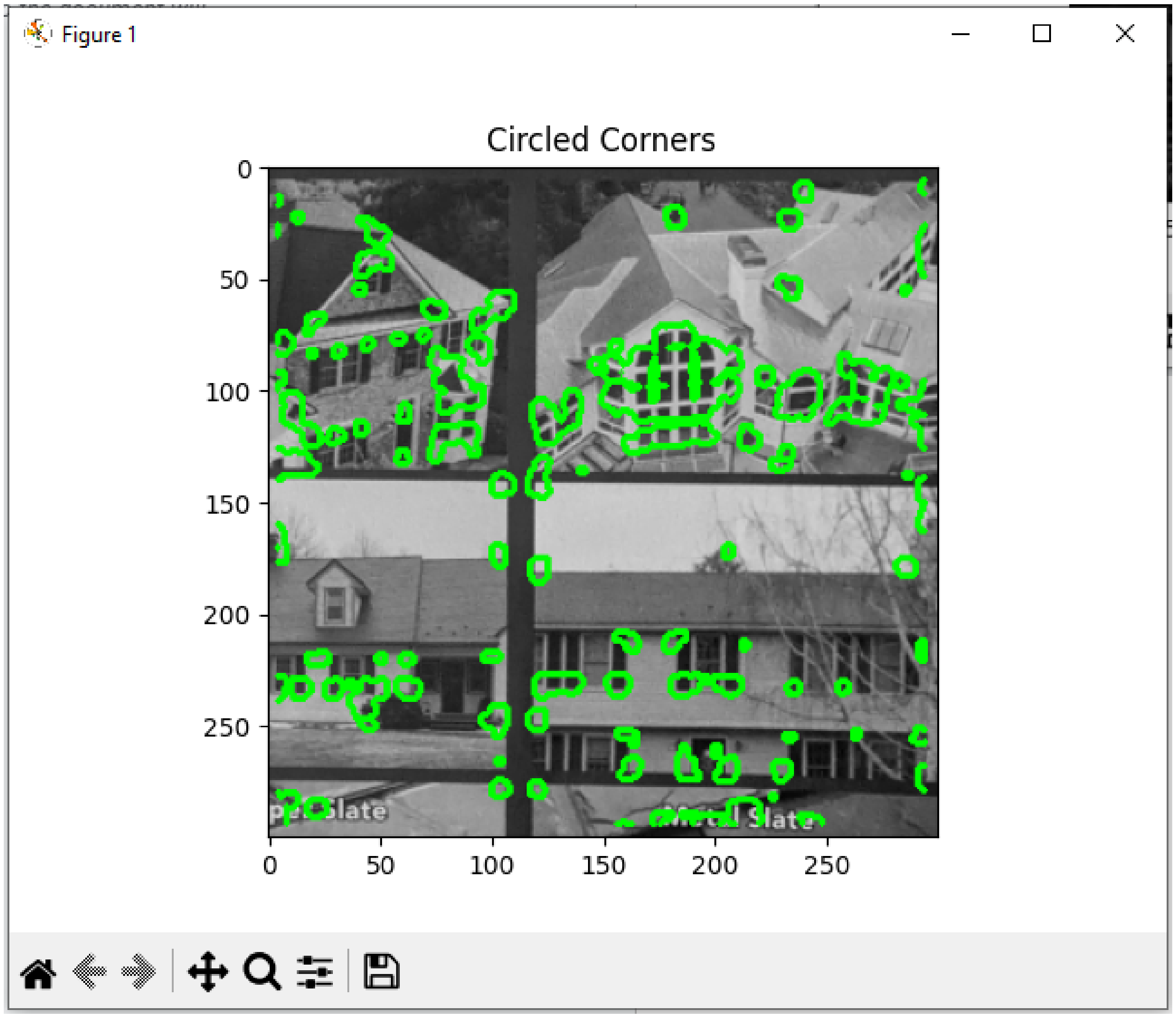
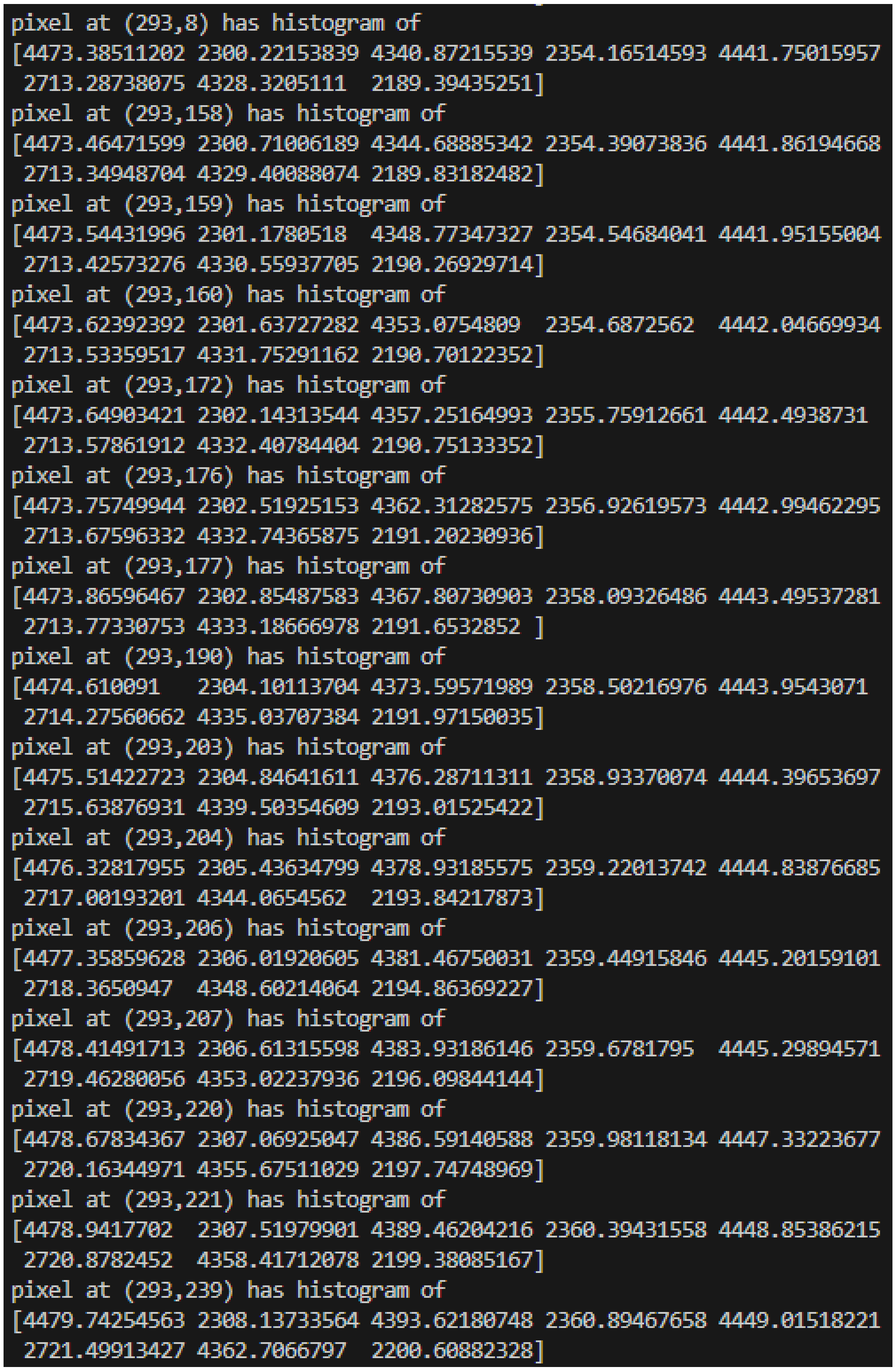


Image of Circled Corners for Part3a



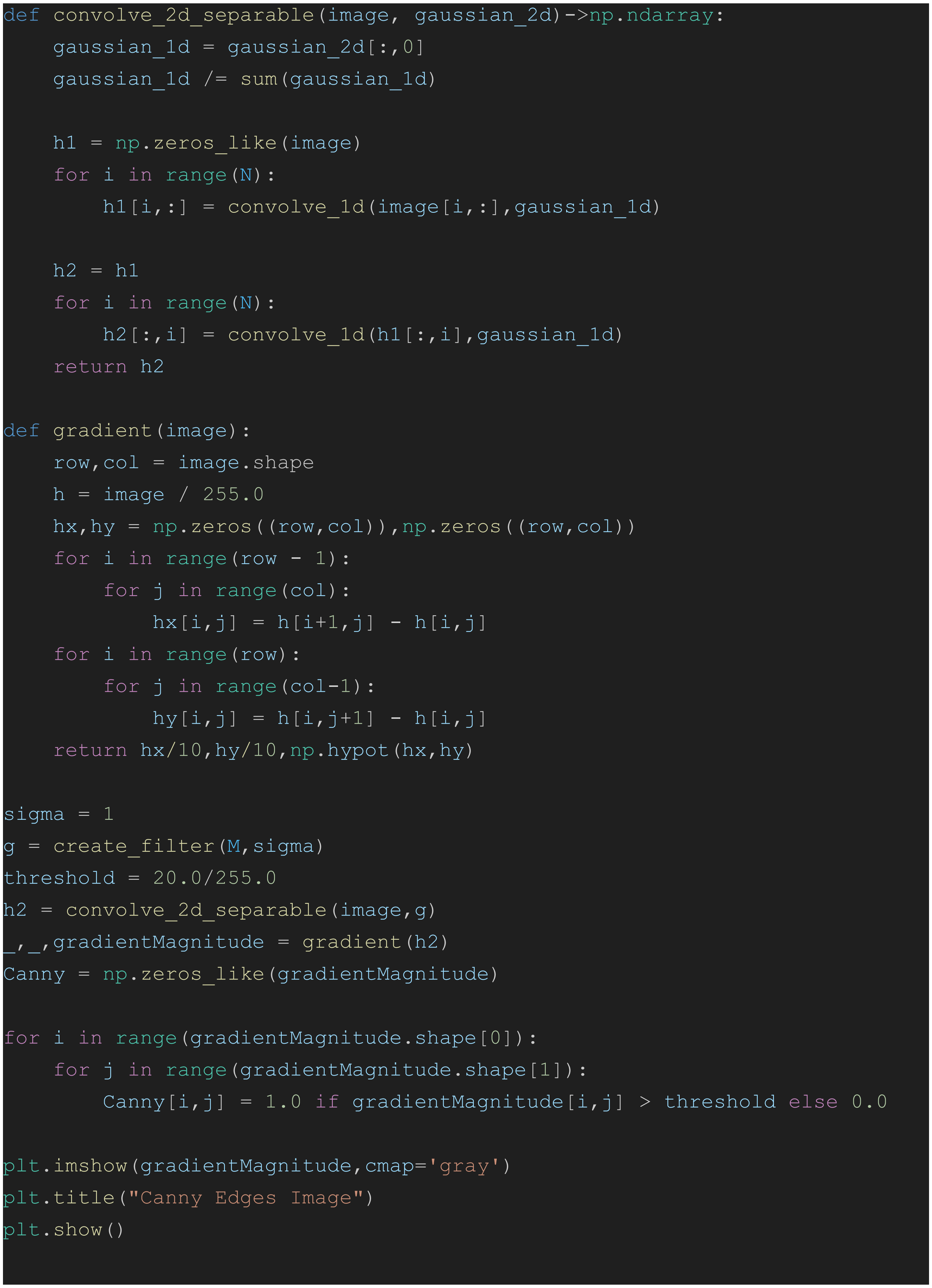
```
pixel at (293,8) has histogram of
[4473.38511202 2300.22153839 4340.87215539 2354.16514593 4441.75015957
 2713.28738075 4328.3205111  2189.39435251]
pixel at (293,158) has histogram of
[4473.46471599 2300.71006189 4344.68885342 2354.39073836 4441.86194668
 2713.34948704 4329.40088074 2189.83182482]
pixel at (293,159) has histogram of
[4473.54431996 2301.1780518  4348.77347327 2354.54684041 4441.95155004
 2713.42573276 4330.55937705 2190.26929714]
pixel at (293,160) has histogram of
[4473.62392392 2301.63727282 4353.0754809  2354.6872562  4442.04669934
 2713.53359517 4331.75291162 2190.70122352]
pixel at (293,172) has histogram of
[4473.64903421 2302.14313544 4357.25164993 2355.75912661 4442.4938731
 2713.57861912 4332.40784404 2190.75133352]
pixel at (293,176) has histogram of
[4473.75749944 2302.51925153 4362.31282575 2356.92619573 4442.99462295
 2713.67596332 4332.74365875 2191.20230936]
pixel at (293,177) has histogram of
[4473.86596467 2302.85487583 4367.80730903 2358.09326486 4443.49537281
 2713.77330753 4333.18666978 2191.6532852 ]
pixel at (293,190) has histogram of
[4474.610091   2304.10113704 4373.59571989 2358.50216976 4443.9543071
 2714.27560662 4335.03707384 2191.97150035]
pixel at (293,203) has histogram of
[4475.51422723 2304.84641611 4376.28711311 2358.93370074 4444.39653697
 2715.63876931 4339.50354609 2193.01525422]
pixel at (293,204) has histogram of
[4476.32817955 2305.43634799 4378.93185575 2359.22013742 4444.83876685
 2717.00193201 4344.0654562  2193.84217873]
pixel at (293,206) has histogram of
[4477.35859628 2306.01920605 4381.46750031 2359.44915846 4445.20159101
 2718.3650947  4348.60214064 2194.86369227]
pixel at (293,207) has histogram of
[4478.41491713 2306.61315598 4383.93186146 2359.6781795  4445.29894571
 2719.46280056 4353.02237936 2196.09844144]
pixel at (293,220) has histogram of
[4478.67834367 2307.06925047 4386.59140588 2359.98118134 4447.33223677
 2720.16344971 4355.67511029 2197.74748969]
pixel at (293,221) has histogram of
[4478.9417702  2307.51979901 4389.46204216 2360.39431558 4448.85386215
 2720.8782452  4358.41712078 2199.38085167]
pixel at (293,239) has histogram of
[4479.74254563 2308.13733564 4393.62180748 2360.89467658 4449.01518221
 2721.49913427 4362.7066797  2200.60882328]
```

Printed Results for Part3b

Source Code:

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

###########
# PART 1
###########

#read image
image = plt.imread("pic1grey300.jpg",0)
# image = plt.imread("pic1grey300.jpg",0)
plt.imshow(image,cmap='gray')
plt.title("Original Image")
plt.show()
M = 9
N = image.shape[0]


def gaussian(size, sigma):
    ax = [i - ((size - 1) / 2) for i in range(size)]
    coefficient = np.exp(-0.5 * np.square(ax) / np.square(sigma))
    gaussian = np.outer(coefficient, coefficient)
    return gaussian / gaussian.sum()

#create filter based on specified file or generate a gaussian filter
def create_filter(M, sigma=None, filter_file=None):
    if filter_file:
        # Read filter coefficients from a specified file
        g = np.loadtxt(filter_file)
    else:
        # Create a Gaussian filter with the given sigma
        if sigma is None:
            sigma = M / 4.0
        g = gaussian(M, sigma)
    return g

def convolve_2d(image, kernel):
    row, col = image.shape
    offset = (M - 1) // 2
```

```python
    output = np.zeros((row, col))

    for i in range(offset,row-offset):
        for j in range(offset,col-offset):
            for k in range(M):
                for l in range(M):
                    x = i - (k - offset)
                    y = j - (l - offset)

                    if 0<=x < row and 0<=y < col:
                        output [i,j]+=kernel[k,l]*image[x,y]
    return output

# Read filter from textfile
# g = create_filter(M, filter_file="filter.txt")

#create a Gaussian filter
sigma = M/4.0
g = create_filter(M, sigma)

h = convolve_2d(image, g)

# display image for part 1
plt.imshow(h,cmap='gray')
plt.title("Convolved Image")
plt.show()

###########
# PART 2
###########

def convolve_1d(image, kernel):
    size = len(kernel)
    border = (size-1)//2
    output = np.copy(image)

    for i in range(border, len(image)-border):
        output[i] = np.sum(image[i-border:i+border+1] * kernel)
    return output
```



```python
def convolve_2d_separable(image, gaussian_2d)->np.ndarray:
    gaussian_1d = gaussian_2d[:,0]
    gaussian_1d /= sum(gaussian_1d)

    h1 = np.zeros_like(image)
    for i in range(N):
        h1[i,:] = convolve_1d(image[i,:],gaussian_1d)

    h2 = h1
    for i in range(N):
        h2[:,i] = convolve_1d(h1[:,i],gaussian_1d)
    return h2

def gradient(image):
    row,col = image.shape
    h = image / 255.0
    hx,hy = np.zeros((row,col)),np.zeros((row,col))
    for i in range(row - 1):
        for j in range(col):
            hx[i,j] = h[i+1,j] - h[i,j]
    for i in range(row):
        for j in range(col-1):
            hy[i,j] = h[i,j+1] - h[i,j]
    return hx/10,hy/10,np.hypot(hx,hy)

sigma = 1
g = create_filter(M,sigma)
threshold = 20.0/255.0
h2 = convolve_2d_separable(image,g)
_,_,gradientMagnitude = gradient(h2)
Canny = np.zeros_like(gradientMagnitude)

for i in range(gradientMagnitude.shape[0]):
    for j in range(gradientMagnitude.shape[1]):
        Canny[i,j] = 1.0 if gradientMagnitude[i,j] > threshold else 0.0

plt.imshow(gradientMagnitude,cmap='gray')
plt.title("Canny Edges Image")
plt.show()
```

```
###########
# PART 3a
###########

g = create_filter(M,sigma=2)
convolved = convolve_2d_separable(image,g)

Ix, Iy, _ = gradient(convolved)
A = np.zeros((N,N))
B = np.zeros((N,N))
C = np.zeros((N,N))

for i in range(N):
    for j in range(N):
        A[i,j] = np.square(Ix[i,j])
        B[i,j] = np.square(Iy[i,j])
        C[i,j] = Ix[i,j] * Iy[i,j]

smooth=[]
for x in [A,B,C]:
    g = create_filter(9,5)
    smooth.append(convolve_2d_separable(x,g))
Aprime, Bprime, Cprime = smooth

matrix = np.zeros((N,N,2,2))
matrix[:, :, 0, 0] = Aprime
matrix[:, :, 0, 1] = Cprime
matrix[:, :, 1, 0] = Cprime
matrix[:, :, 1, 1] = Bprime

def cornerness(matrix,alpha):
    R = np.zeros((N,N))
    for i in range(N):
        for j in range(N):
            R[i,j] = np.linalg.det(matrix[i,j]) - 0.04 *
np.square((np.trace(matrix[i,j])))

    threshold = alpha*np.max(np.abs(R))

    for i in range(N):
        for j in range(N):
            R[i,j] = 1.0 if R[i,j] > threshold else 0.0
    return R
R = cornerness(matrix,0.001)

def nonmaxima_suppresion(R):
    row,col = R.shape
    corners = np.zeros((row,col))
    for i in range(1,row-1):
        for j in range(1,col-1):
            neighbors = R[i-1:i+2, j-1:j+2]
            if R[i,j] == np.max(neighbors):
                corners[i,j] = 1
    return corners
corners = nonmaxima_suppresion(R)

color = cv2.cvtColor(image,cv2.COLOR_GRAY2BGR)
for i in range(4,N-5):
    for j in range(4,N-5):
        if corners[i,j] == 0:
            cv2.circle(color, (j,i), radius=1, color=(0, 255, 0))
plt.imshow(color)
plt.title("Circled Corners")
plt.show()

###########
# PART 3b
###########

hist = np.array([0,0,0,0,0,0,0,0],dtype=float)
itr = 0
gradient_vector = np.arctan2(Ix,Iy)
gradient_vector = gradient_vector * 180 / np.pi
for i in range(5,N-6):
    for j in range(5,N-6):
        if corners[i,j] == 0:
            for k in range(i-5,i+6):
                for l in range(j-5,j+6):
                    hist[round(gradient_vector[k,l]/45)] +=
gradientMagnitude[k,l]
```

```python
        maxval = 0
        for o in range(len(hist)):
            if hist[o] > maxval:
                maxval = hist[o]
                itr = o
        histn = np.roll(hist,itr)
        print("pixel at (" + str(i) +","+str(j)+") has histogram of ")
        print(histn)
```