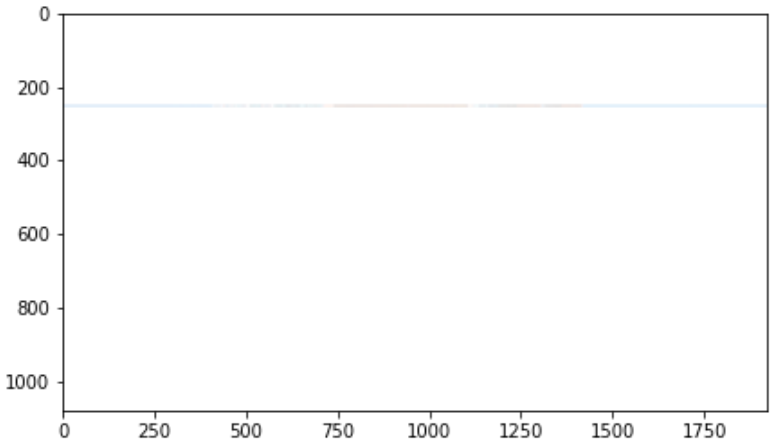
CMSC 733 HW0

UID: 118339238 Name: Young-Shiuan Hsu

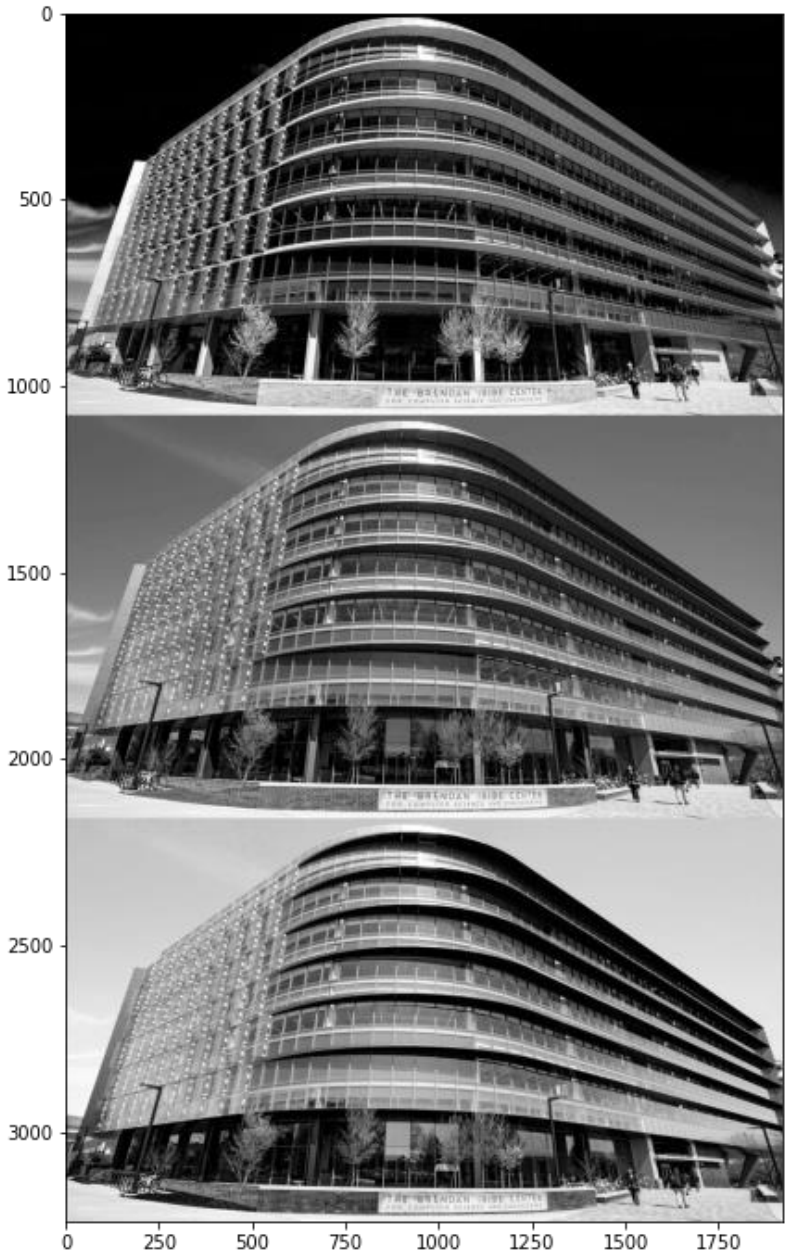
1. Plot the RGB values along the scanline on the 250th row of the image.

```
temp = np.zeros((1,img.shape[1]))
newimg = np.ones((img.shape))
newimg = newimg * 255  ## set the image to white(blank canvas)
for i in range(img.shape[0]):
    if i == 250:
        for j in range(img.shape[1]):
            for k in range(img.shape[2]):
                newimg[i,j,k] = img[i,j,k]

figure(figsize=(7,12))
plt.imshow(newimg.astype('uint8'))
plt.savefig('1_scanline.png')
```



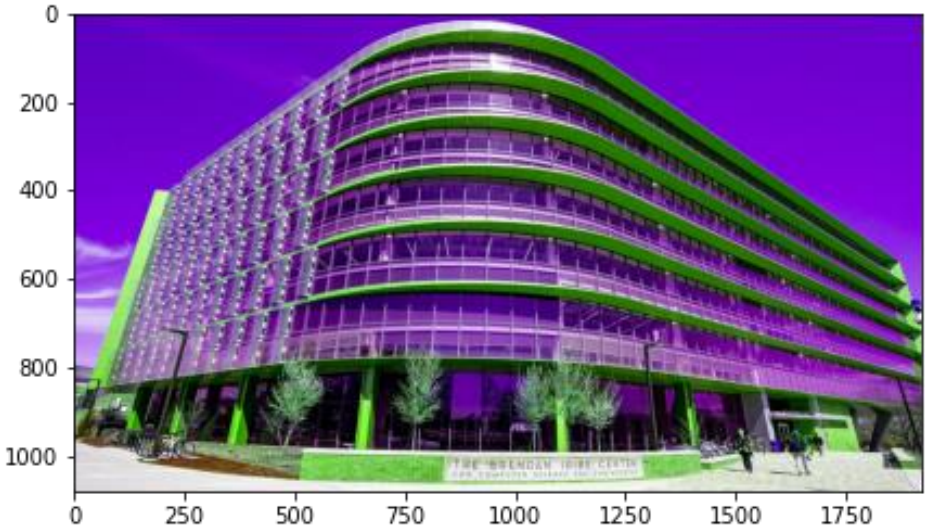
2. Stack the RGB channels of the image vertically

```
imgR = img[:,:,0]  ## separate the image into R
imgG = img[:,:,1]  ## separate the image into G
imgB = img[:,:,2]  ## separate the image into B

newimg = []
for i in range(3):
    for j in range(imgR.shape[0]):
        newimg.append(img[j,:,i])

newimg = np.array(newimg)
figure(figsize=(7,12))
plt.imshow(newimg, cmap=plt.get_cmap('gray'))
plt.savefig('2_concat.png')
```



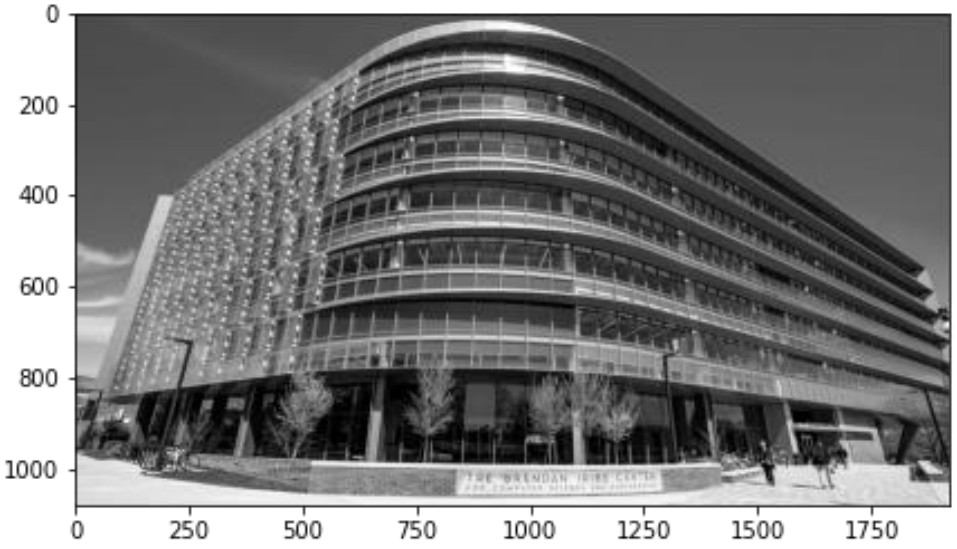
3. Swap red and green color channel

```
swapimg = np.zeros(img.shape)
swapimg[:,:,0] = imgG
swapimg[:,:,1] = imgR
swapimg[:,:,2] = imgB

figure(figsize=(7,12))
plt.imshow(swapimg.astype('uint8'))
plt.savefig('3_swapchannel.png')
```



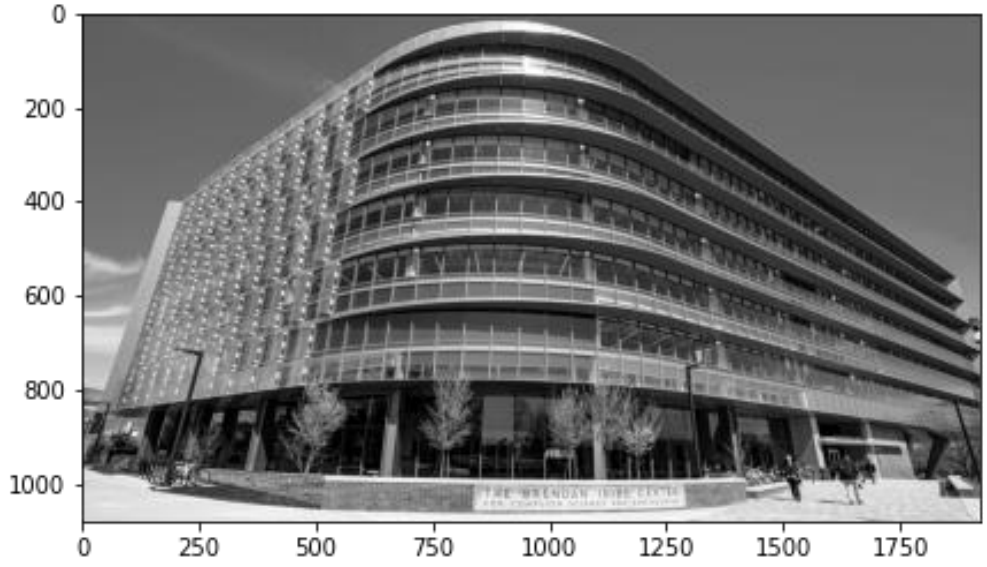
4. Load the input color image to the grayscale image
   We use a certain formula to convert an RGB image into a gray scale image.
   New grayscale image = 0.3*R + 0.59*G + 0.11*B
   Reference website:
   https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm#:~:text=You%20just%20have%20to%20take,Its%20done%20in%20this%20way

```
'''
Project RGB values onto a vector ie.assumes the distribution of the three channe
New grayscale image = (0.3 * R) + (0.59 * G) + (0.11 * B)
From this equation we notice that human eyes are more sensitive to green, which
reference website:
https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm#:~:text=You%2
'''

def rgb2gray(img):
    row = img.shape[0]
    col = img.shape[1]
    channel = 3
    newimg = np.zeros((row,col))
    for i in range(row):
        for j in range(col):
            newimg[i,j] = 0.3*img[i,j,0] + 0.59*img[i,j,1] + 0.11*img[i,j,2]
    return newimg


gray = rgb2gray(img)
figure(figsize=(7,12))
plt.imshow(gray,cmap=plt.get_cmap('gray'))
plt.savefig('4_grayscale.png')
```



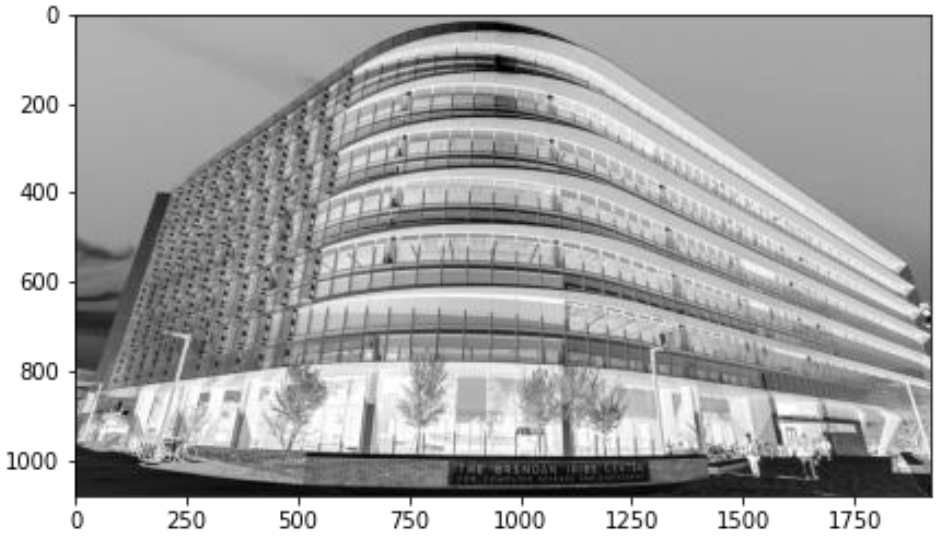
5. Compute the average value of the RGB values over the three channels

```
newimg = np.zeros((row,col))

## method 1 is to take the average of the three channels ie.a
for i in range(row):
    for j in range(col):
        temp = 0
        for k in range(channel):
            temp += img[i,j,k]/channel
        newimg[i,j] = temp

figure(figsize=(7,12))
plt.imshow(newimg.astype('uint8'),cmap=plt.get_cmap('gray'))
plt.savefig('5_average.png')
```



6. Negative image of the grayscale image from problem 4

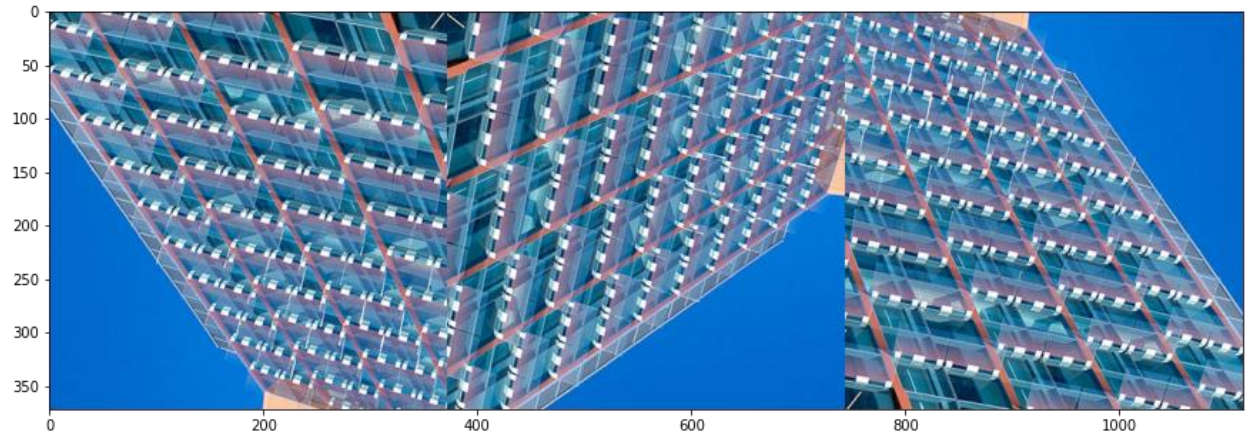
```
newimg = np.zeros((row,col))
for i in range(row):
    for j in range(col):
        newimg[i,j] = 255 - gray[i,j]

figure(figsize=(7,12))
plt.imshow(newimg,cmap=plt.get_cmap('gray'))
plt.savefig('6_negative.png')
```



7. Crop the image to 372*372, then rotate the image by 90,180 and 270 degrees

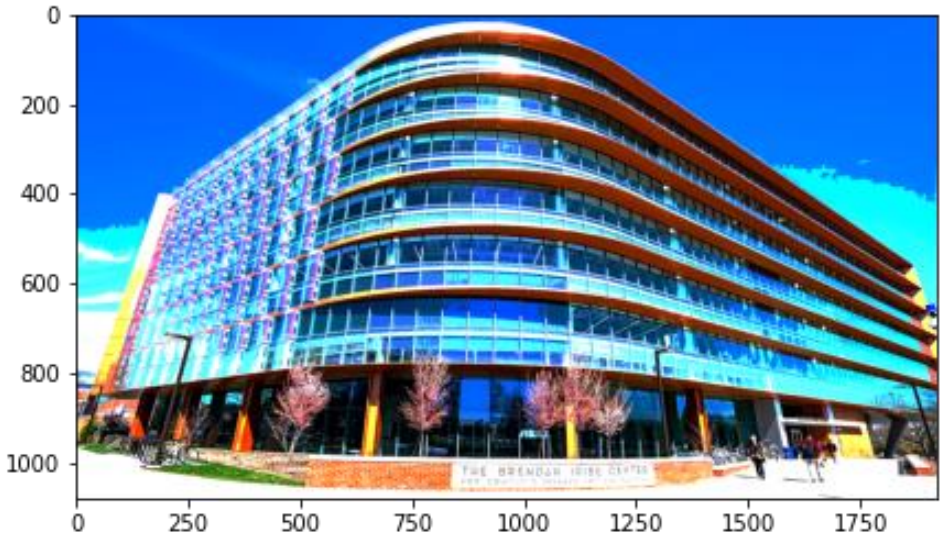
```
def rotate_matrix90(img):  ## rotate the matrix by 90 degrees
    row = img.shape[0]
    col = img.shape[1]
    channels = img.shape[2]
    newimg = np.zeros((col,row, channels))
    for i in range(row):
        for j in range(col):
            for k in range(channels):
                newimg[i,j,k] = img[j,row-i-1,k]
    return newimg
def rotate_matrix180(img):
    row = img.shape[0]
    col = img.shape[1]
    channels = img.shape[2]
    newimg = np.zeros((row, col, channels))
    for i in range(row):
        for j in range(col):
            for k in range(channels):
                newimg[i,j,:] = img[row-i-1,col-j-1,:]
    return newimg

def rotate_matrix270(img):
    row = img.shape[0]
    col = img.shape[1]
    channels = img.shape[2]
    newimg = np.zeros((col, row, channels))
    for i in range(row):
        for j in range(col):
            for k in range(channels):
                newimg[i,j,:] = img[col-j-1,i,:]
    return newimg
```

```
rotate1 = rotate_matrix90(crop_img) ## rotate image by 90 degrees
rotate2 = rotate_matrix180(crop_img) ## rotate image by 180 degrees
rotate3 = rotate_matrix270(crop_img) ## rotate image by 270 degrees

newimg = np.zeros((372, 372*3, 3))
for i in range(372):
    newimg[:,i,:] = rotate1[:,i,:]
for i in range(372):
    newimg[:,i+372,:] = rotate2[:,i,:]
for i in range(372):
    newimg[:,i+372*2,:] = rotate3[:,i,:]

figure(figsize=(15,9))
plt.imshow(newimg.astype('uint8'))
plt.savefig('7_rotation.png')
```



8. Set the pixel values that are greater than 128 to 255

```
maskimg = np.zeros(img.shape)
for i in range(row):
    for j in range(col):
        for k in range(channel):
            if(img[i,j,k] > 127):
                maskimg[i,j,k] = 255
            else:
                maskimg[i,j,k] = img[i,j,k]
figure(figsize=(7,12))
plt.imshow(maskimg.astype('uint8'))
plt.savefig('8_mask.png')
```



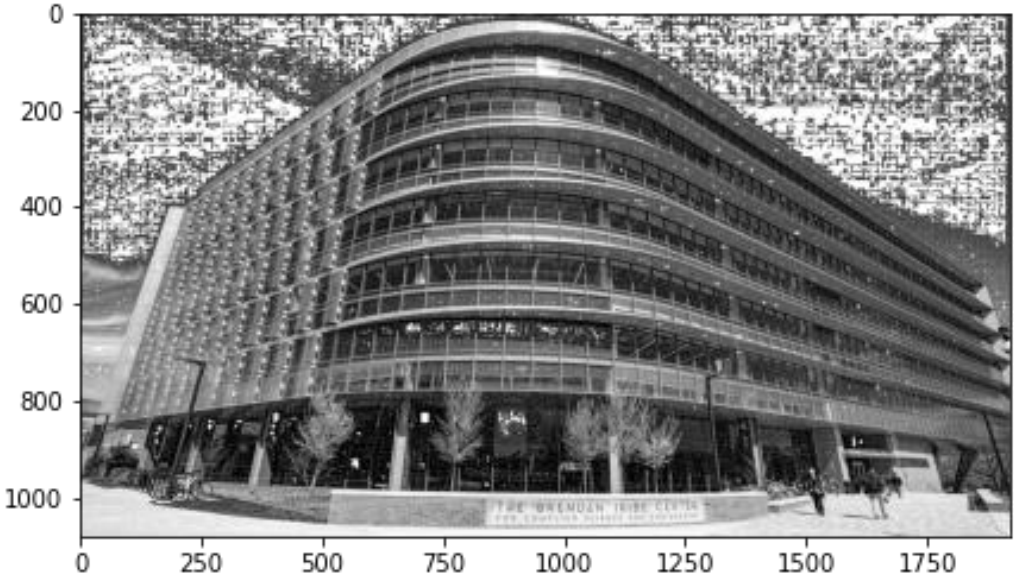
9. The mean of the RGB values of the masked image in problem 8

```
Rmean = 0
Gmean = 0
Bmean = 0

for i in range(row):
    for j in range(col):
        Rmean += maskimg[i,j,0]
        Gmean += maskimg[i,j,1]
        Bmean += maskimg[i,j,2]
Rmean = Rmean / (row*col)
Gmean = Gmean / (row*col)
Bmean = Bmean / (row*col)

print("R mean:", Rmean)
print("G mean:", Gmean)
print("B mean:", Bmean)
```

```
R mean: 85.71435185185184
G mean: 129.462589216821
B mean: 170.22854166666667
```

10. For every 5*5 patch, find the maximum value in the patch and set it to 255
    Note: if there are multiple pixels in the 5*5 kernel that is equivalent to the maximum value we set them all to 255

```
def find_max_reset(matrix):
    maxval = 0
    X = 0
    Y = 0
    for i in range(5):
        for j in range(5):
            if(matrix[i][j] >= maxval):
                maxval = matrix[i][j]
                X = i
                Y = j
    for i in range(5):
        for j in range(5):
            if(matrix[i][j] == maxval):
                matrix[i][j] = 255
    return matrix

def convolve_kernel(img):
    ## a 5*5 kernel
    centerX = 0
    centerY = 0
    patchx = int(img.shape[0] / 5)
    patchy = int(img.shape[1] / 5)
    for i in range(patchx-1):
        for j in range(patchy-1):
            temp = np.ones((5,5))
            for a in range(5):
                for b in range(5):
                    temp[a][b] = img[centerX+a][centerY+b]
            temp = find_max_reset(temp)
            for a in range(5):
                for b in range(5):
                    img[centerX+a][centerY+b] = temp[a][b]
            centerY = centerY+5
        centerX = centerX+5
        centerY = 0


newimg = convolve_kernel(gray)
figure(figsize=(7,12))
plt.imshow(gray.astype('uint8'), cmap=plt.get_cmap('gray'))
plt.savefig('10_nonmax.png')
```


0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750