Harris Corners Report
By: Kareem Tiwari

This Harris Corners program finds the Harris corners using the calculations given in class. After we used the Harris corner equations we then found the Cornerness measurements in the two ways that were described in class and in the videos. Finally, we use the visualization on as many other images aside from the two starter images in the instructions to see how well the cornerness equations work on different images and contrasts.

First the we start by loading the image and creating the empty images for Ix, Iy,IxIx ,IyIy, and IxIy. Then as we loop through the all the pixels in the image we fill in the values for all of the Ix,Iy,IxIx,IyIy, and IxIy values. Then, we create the empty image to store the Cornerness values. Then as we loop through all the pixels of the image again we access all the values in the neighborhood that we would decide. We then sum up all the IxIx, IyIy,IxIy values in the neighborhood, compute the determinant of the matrix of those sums in the M-matrix, compute the trace of the M-matrix as well as the computation of the cornerness value of the center pixel.

```
###
import cv2
import numpy as np

im = cv2.imread("starry_night_full.jpeg",0) # Grayscale image
numRows = im.shape[0] # x value
numCols  = im.shape[1] # y value
emptyim_Ix = np.zeros((numRows,numCols,1),np.float32)#create  an empty image
emptyim_IxIx = np.zeros((numRows,numCols,1),np.float32)#create  an empty image
emptyim_Iy = np.zeros((numRows,numCols,1),np.float32)#create  an empty image
emptyim_IyIy = np.zeros((numRows,numCols,1),np.float32)#create  an empty image
emptyim_IxIy = np.zeros((numRows,numCols,1),np.float32)#create  an empty image
cornerness_im =  np.zeros((numRows,numCols,1),np.float32)
width = numRows/5
height =  numCols/5

# a = i and b = j( when looking at the Harris Corner coding notes on gradient
PixelGrad_Array = [] # create an array to store the gradient values from every pixel
for i in range(1,numRows-1):
    for j in range(1,numCols-1):
        emptyim_Ix[i,j] = int(im[i+1,j]) - int(im[i-1,j]) # gradient x

        emptyim_Iy[i,j] = int(im[i,j+1]) - int(im[i,j-1]) #gradient y

        emptyim_IxIx[i,j]= emptyim_Ix[i,j] *  emptyim_Ix[i,j]

        emptyim_IyIy[i,j] = emptyim_Iy[i,j] * emptyim_Iy[i,j]

        emptyim_IxIy[i,j] = emptyim_Ix[i,j] * emptyim_Iy[i,j]
Max_C=0
for  i  in range(1,numRows-1):
    for j in range(1,numCols-1):
        Sum_IxIx = emptyim_IxIx[i,j]+emptyim_IxIx[i+1,j] +emptyim_IxIx[i-1,j] + emptyim_IxIx[i,j-1] +emptyim_IxIx[i,j+1] +emptyim_IxIx[i-1,j-1]+ emptyim_IxIx[i+1,j+1]+emptyim_Ix
        Sum_IxIy = emptyim_IxIy[i,j]+emptyim_IxIy[i+1,j] +emptyim_IxIy[i-1,j] + emptyim_IxIy[i,j-1] +emptyim_IxIy[i,j+1] +emptyim_IxIy[i-1,j-1]+ emptyim_IxIy[i+1,j+1]+emptyim_Ix
        Sum_IyIy = emptyim_IyIy[i,j]+emptyim_IyIy[i+1,j] +emptyim_IyIy[i-1,j] + emptyim_IyIy[i,j-1] +emptyim_IyIy[i,j+1] +emptyim_IyIy[i-1,j-1]+ emptyim_IyIy[i+1,j+1]+emptyim_Iy

        M_det = (Sum_IxIx*Sum_IyIy) -(Sum_IxIy *Sum_IxIy) #This gets the determinant value for every pixel  in the  image
        M_trace = (Sum_IxIx +Sum_IyIy) #This gets the trace value for every pixel in the image
```

(in the image above all of the empty images are created. The loops to iterate through all of the pixels in the image as well as the gradient calculations. The Trace and the Determinant calculation for the M matrix as well as the Sums of the IxIx IyIy and IxIy values for the M matrix.)

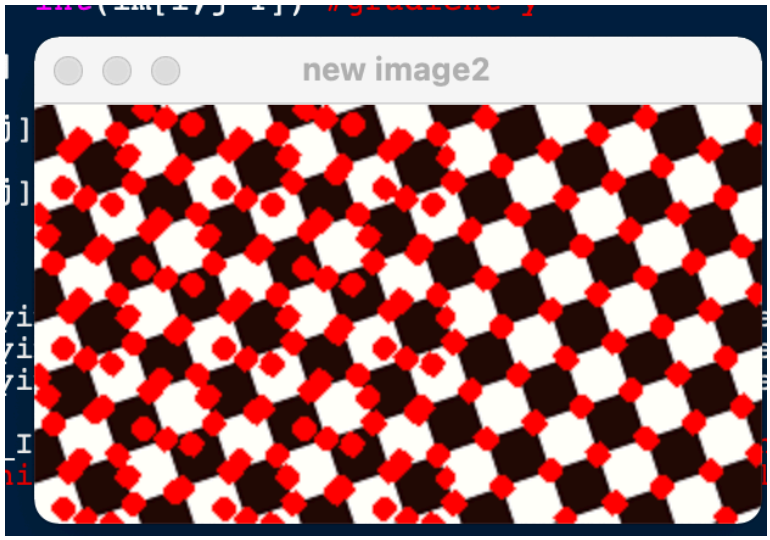
We then have to use the cornerness value that we got from the cornerness equations to find the max cornerness value. We then use that max cornerness value to find the threshold value in which we then can compare the cornerness values in an image at the point (I,j) to the threshold value so that we are able to then find the corners of an image. (below)

```
        C = M_det - (0.05)*(M_trace)**2
        if C > Max_C:
            Max_C = C


        cornerness_im[i,j] = C

threshold = Max_C*(0.10)
im_color = cv2.imread("starry_night_full.jpeg")
for i in range(numRows):
    for j in range(numCols):

        if cornerness_im[i,j] > threshold:
            cv2.circle(im_color,(j,i),1,(0,0,255),4)
```

Then, we also were tasked to find the cornerness values while using blocking and in my attempt I used a 5 by 5 block while iterating through the image to be able to find the corners this way as well. (code below)

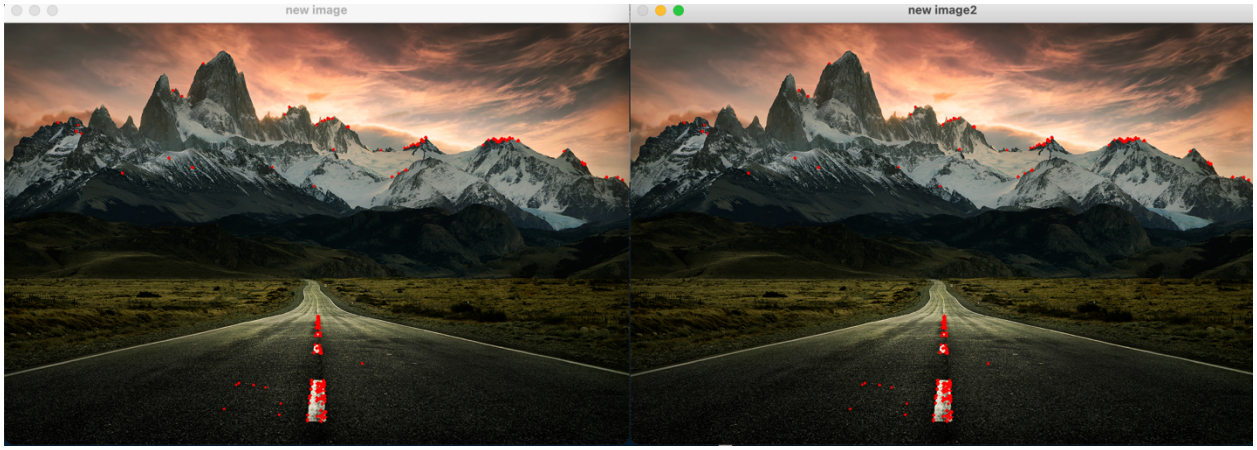
```
for i in range(5):
    for j in range(5):
        for x  in range((i*int(width)),(i*int(width)+int(width))):
            for y  in range(j*int(height),(j*int(height)+int(height))):

                if cornerness_im[x,y]>threshold:
                    cv2.circle(im_color,(y,x),1,(0,0,255),4)

cv2.imshow("new image2",im_color)
```

One of the problems that I ran into while creating the second set of for loops that are finding the corners through blocking is that I had my x and y values confused shifting all of the corners to the left side of the image while the rest of the image had no values spread across the right side of the image.(below)

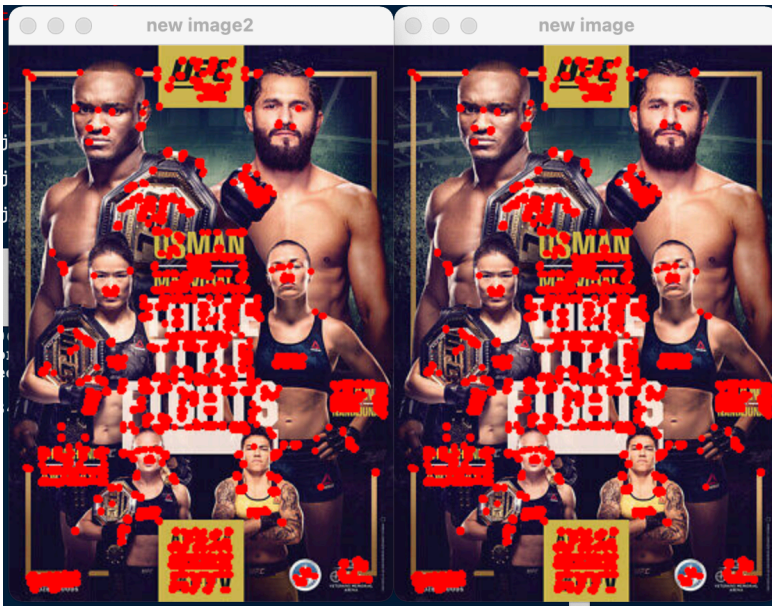


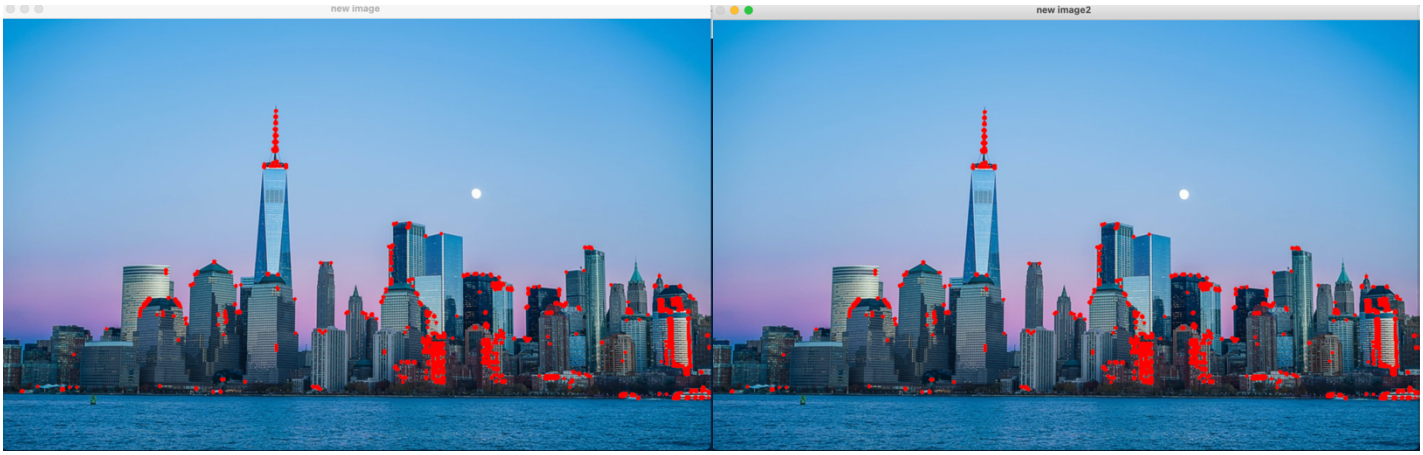
Finally, Here are the five images that I found all of the corners on.


new image
new image2


new image
new image2


new image2
new image


new image2
new image
USMAN
FIGHTS


new image
new image2