CSE 5524 HW8 Utkarsh Pratap Singh Jadon

Question 1

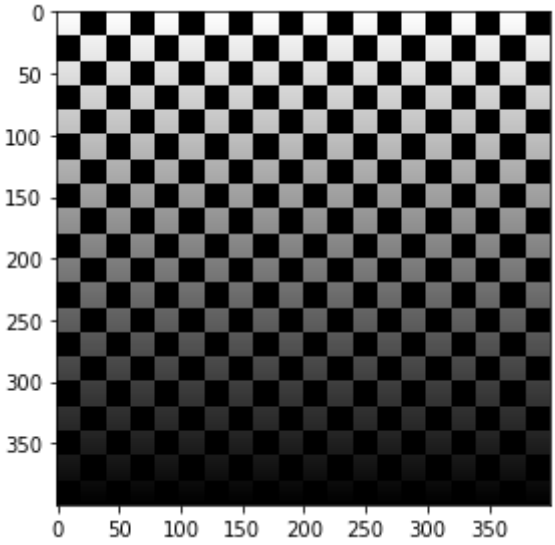
Import necessary libraries

In [437…
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from skimage.color import rgb2gray
import cv2 as cv
import numpy as np
import math
from PIL import Image
import glob
import os
import skimage
import scipy.ndimage
from os import listdir
from os.path import join, isfile
from skimage import morphology
from skimage import measure,color
from skimage import io, data
from numpy.linalg import eig
from scipy import ndimage, misc
from scipy.ndimage import median_filter
import matplotlib.patches as patches
```

Read and display input image

In [500…
```
checkerImage = skimage.io.imread('checker.png')
skimage.io.imshow(checkerImage)
```

Out[500]:
```
<matplotlib.image.AxesImage at 0x7fa027d104c0>
```



Write a function to compute 2D Gaussian derivative masks Gx and Gy with standard deviation (sigmaD) of 0.7

In [525…

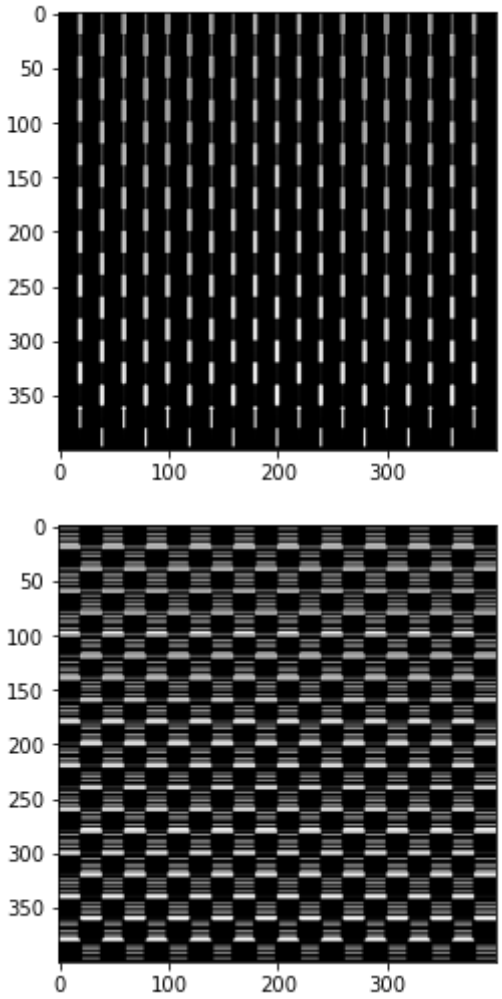
```
def gaussDeriv2D(sigmaD):
    x=np.arange(-math.ceil(3*sigmaD),math.ceil(3*sigmaD),1)
    y=np.arange(-math.ceil(3*sigmaD),math.ceil(3*sigmaD),1)
    factor = 1/(2*math.pi*math.pow(sigmaD,4))
    Gx=np.zeros((y.size,x.size))
    Gy=np.zeros((y.size,x.size))
    for j in range(0,y.size,1):
        for i in range(0,x.size,1):
            Gx[j][i]=x[i]*np.exp(-(x[i]*x[i]+y[j]*y[j])/(2*sigmaD**2))*factor
    Gx = Gx / (np.sum(np.abs(Gx)))
    for j in range(0,y.size,1):
        for i in range(0,x.size,1):
            Gy[j][i]=y[j]*np.exp(-(x[i]*x[i]+y[j]*y[j])/(2*sigmaD**2))*factor
    Gy = Gy / (np.sum(np.abs(Gy)))
    return [Gx, Gy]

[Gx,Gy]=gaussDeriv2D(0.7)
print(np.sum(np.abs(Gx)))
print(np.sum(np.abs(Gy)))
```

```
0.9999999999999998
0.9999999999999999
```

Apply gaussian derivate to get Image derivatives (Ix, Iy)

In [526…

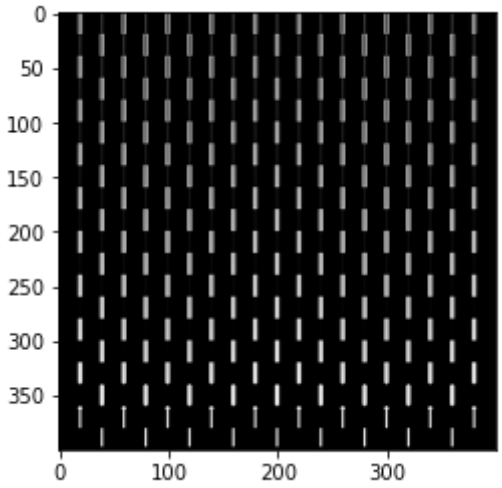
```
# Calulcate Ix

Ix=scipy.ndimage.correlate(checkerImage, Gx, mode='nearest')

# Calculate Iy

Iy=scipy.ndimage.correlate(checkerImage, Gy, mode='nearest')

plt.subplot(1,1,1)
plt.imshow(Ix, cmap='gray')
plt.show()

plt.subplot(1,1,1)
plt.imshow(Iy, cmap='gray')
plt.show()
```

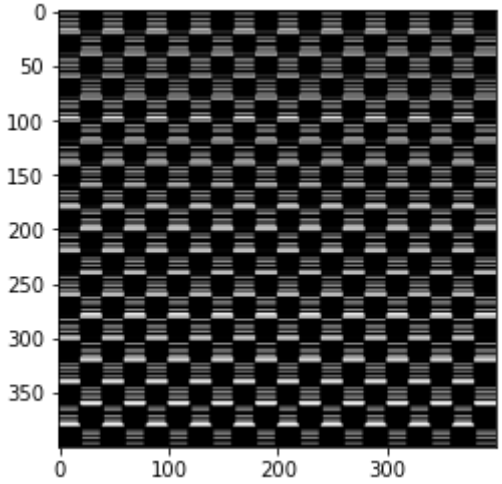


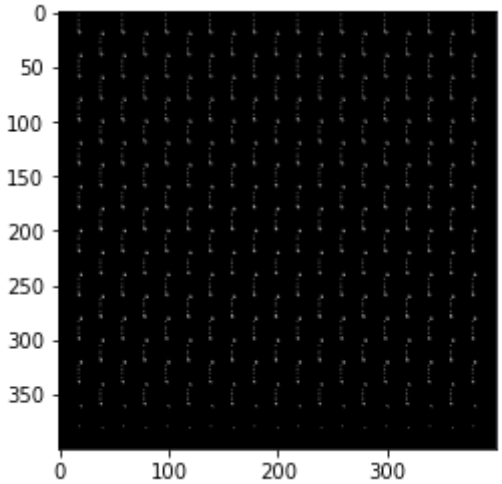
Multiply derivates to get Ix^2, Iy^2, and IxIy

In [527…

```
IxSquare = np.square(Ix,dtype='long')
IySquare = np.square(Iy,dtype='long')
IxIy = np.multiply(Ix,Iy,dtype='long')

plt.subplot(1,1,1)
plt.imshow(IxSquare, cmap='gray')
plt.show()

plt.subplot(1,1,1)
plt.imshow(IySquare, cmap='gray')
plt.show()

plt.subplot(1,1,1)
plt.imshow(IxIy, cmap='gray')
plt.show()
```





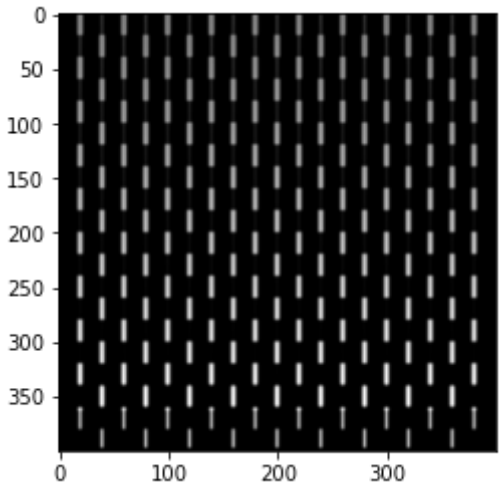


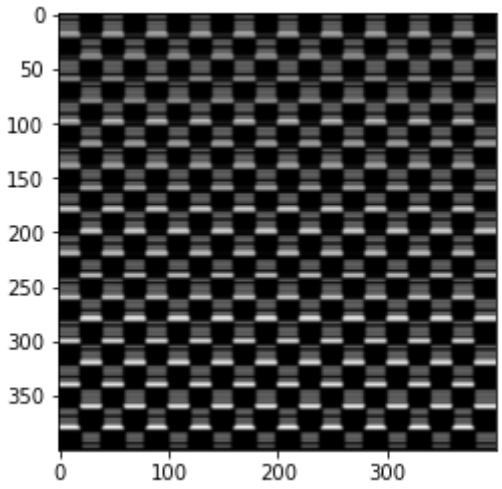
Apply gaussian blur with standard deviation (sigmaI) of 1

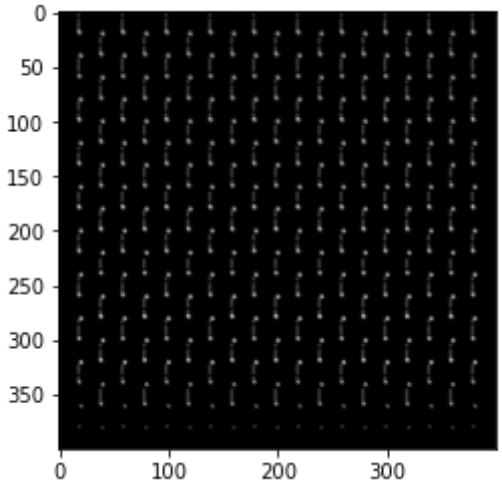
In [528…

```
sigmaI = 1
gaussianBlurIxSquare = cv.GaussianBlur(np.float32(IxSquare),(2*math.ceil(3*sigm
gaussianBlurIySquare = cv.GaussianBlur(np.float32(IySquare),(2*math.ceil(3*sigm
gaussianBlurIxIy = cv.GaussianBlur(np.float32(IxIy),(2*math.ceil(3*sigmaI)+1,2*

plt.subplot(1,1,1)
plt.imshow(gaussianBlurIxSquare, cmap='gray')
plt.show()
```

```
plt.subplot(1,1,1)
plt.imshow(gaussianBlurIySquare, cmap='gray')
plt.show()

plt.subplot(1,1,1)
plt.imshow(gaussianBlurIxIy, cmap='gray')
plt.show()
```





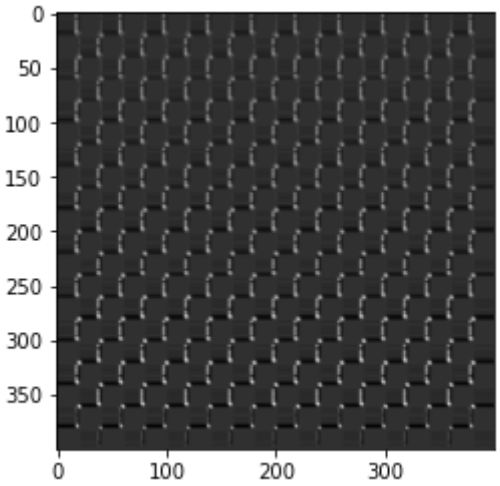


Computer Cornerness function R with trace weighting factor (alpha) = 0.05

In [529…
```
alpha = 0.05
R = (gaussianBlurIxSquare * gaussianBlurIySquare) - (gaussianBlurIxIy) ** 2 - (
```

Display R and values of R(17:23, 17:23)

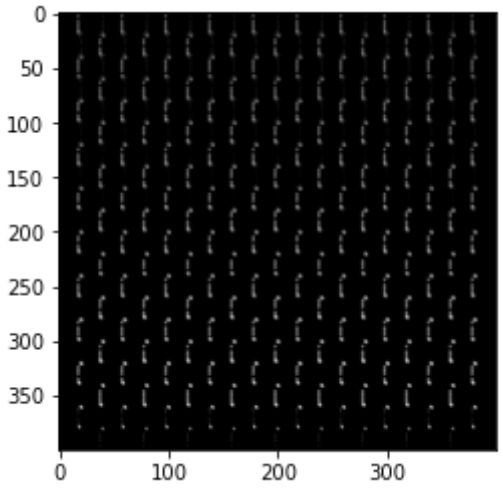
In [530…
```
plt.subplot(1,1,1)
plt.imshow(R, cmap='gray')
plt.show()

print(R[16:22,16:22])
```




```
[[ 4.04992440e+07  2.38356672e+08  3.02212448e+08  1.33071960e+08
   1.16225240e+07 -3.52126720e+07]
 [ 3.26499680e+07  3.63099712e+08  4.33404384e+08  2.53913008e+08
   1.09485000e+08 -5.37048000e+05]
 [ 2.01048160e+07  3.35519040e+08  3.39197696e+08  2.19474768e+08
   1.74472512e+08  6.77824320e+07]
 [ 1.38010320e+07  2.03818256e+08  2.42288016e+08  1.57131184e+08
   1.37595504e+08  1.02873656e+08]
 [-1.63927320e+07  8.25778720e+07  1.67861360e+08  1.37275056e+08
   9.36155120e+07  6.09031520e+07]
 [-3.76614080e+07 -3.34256400e+06  6.67438920e+07  1.02825248e+08
   6.09152960e+07  1.82537420e+07]]
```

Display thresholded R and thresholded values of R(17:23, 17:23)

In [531…
```
threshold = 1000000
thresholdedR = np.where(R >= threshold, R, 0)
plt.subplot(1,1,1)
plt.imshow(thresholdedR, cmap='gray')
plt.show()

print(thresholdedR[16:22,16:22])
```



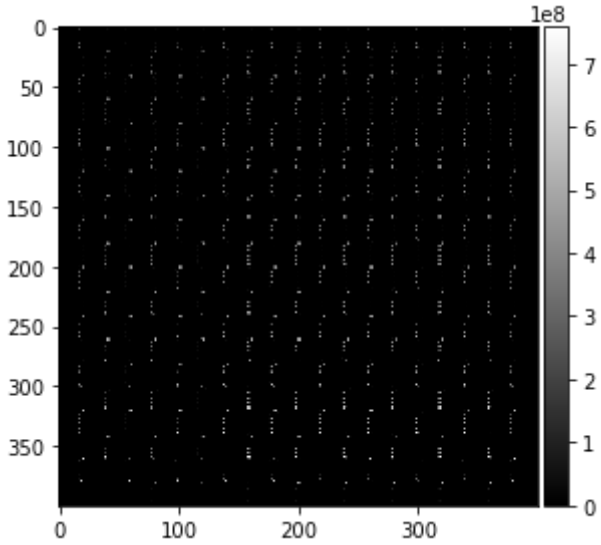

```
[[4.04992440e+07 2.38356672e+08 3.02212448e+08 1.33071960e+08
  1.16225240e+07 0.00000000e+00]
 [3.26499680e+07 3.63099712e+08 4.33404384e+08 2.53913008e+08
  1.09485000e+08 0.00000000e+00]
 [2.01048160e+07 3.35519040e+08 3.39197696e+08 2.19474768e+08
  1.74472512e+08 6.77824320e+07]
 [1.38010320e+07 2.03818256e+08 2.42288016e+08 1.57131184e+08
  1.37595504e+08 1.02873656e+08]
 [0.00000000e+00 8.25778720e+07 1.67861360e+08 1.37275056e+08
  9.36155120e+07 6.09031520e+07]
 [0.00000000e+00 0.00000000e+00 6.67438920e+07 1.02825248e+08
  6.09152960e+07 1.82537420e+07]]
```

Perform non-maximum suppression on R

In [532…

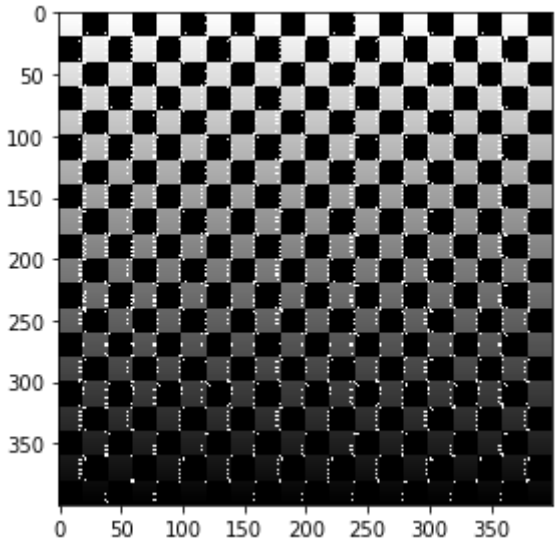
```
thresholdedSuppressedR = thresholdedR
a,b=thresholdedSuppressedR.shape

for i in range(0,a-2,3):
    for j in range(0,b-2,3):
        maxValue = np.max(thresholdedSuppressedR[i:i+3,j:j+3])
        count = 0
        for k in range(i,i+3):
            for l in range(j,j+3):
                if(thresholdedSuppressedR[k][l] < maxValue):
                    thresholdedSuppressedR[k][l] = 0
                else:
                    count += 1
                    if(count > 1):
                        thresholdedSuppressedR[k][l] = 0
io.imshow(thresholdedSuppressedR,cmap='gray')
```

Out[532]: <matplotlib.image.AxesImage at 0x7fa03604f0a0>



In [533…
```
image = checkerImage
image = np.where(thresholdedSuppressedR,255,image)
io.imshow(image)
```

Out[533]:
```
<matplotlib.image.AxesImage at 0x7fa075662190>
```



Discussion

I used gaussian derivative with sigma = 0.7 and gaussian smoothing with sigma = 1 to generate cornerness function R with trace weighting factor as 0.05. All the values less than 1,000,000 were removed by threshholding and non-maximum suppression was performed. Through this, we can extract corners and infer features of an image.

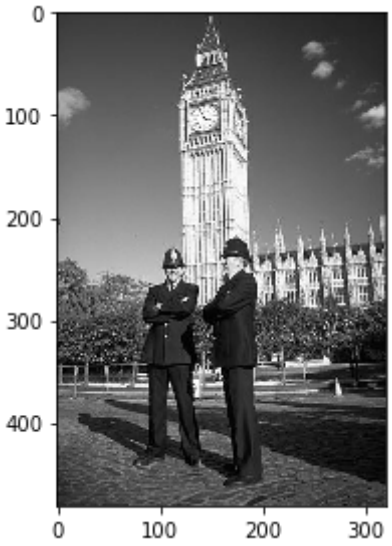
Question 2

Read and display input image

In [156…
```
towerImage = skimage.io.imread('tower.png')
```

```
skimage.io.imshow(towerImage)
print(towerImage.shape)
```

```
(481, 321)
```



Write a function to store intensities of 16 neighbor pixel locations on circular border of radius 3 for all pixels

In [433…

```
def circularNeighborIntensities(img):
    a,b = img.shape
    circularPixelIntensities = np.zeros((a,b,16))

    for i in range(3,a-3):
        for j in range(3,b-3):
            circularPixelIntensities[i][j] = img[i-3,j],img[i-3,j+1],img[i-2,j+

    return circularPixelIntensities
```

Call function and store 16 intensity values for all possible circular border pixels

In [436…

```
neighborIntensities = circularNeighborIntensities(towerImage)
```

Create Contiguous list for threshhold T = 10

In [513…

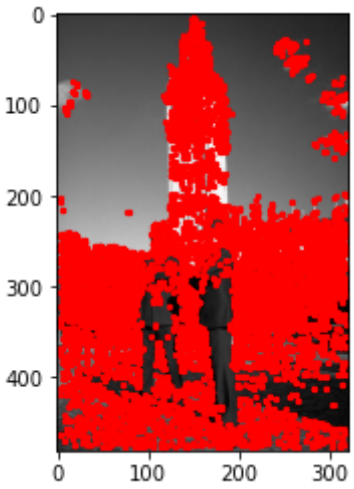
```
T = 10
a,b = towerImage.shape
contiguousList=np.zeros((a,b,16))
for i in range(3,a-3):
    for j in range(3,b-3):
        for k in range(16):
            if(neighborIntensities[i][j][k] > towerImage[i][j] + T):
                contiguousList[i][j][k] = 1
            elif(neighborIntensities[i][j][k] < towerImage[i][j] - T):
                contiguousList[i][j][k] = -1
```

Detect FAST feature points for threshhold T = 10

In [514…
```
fast = np.zeros((a,b))
fastX = []
fastY = []
nStar = 9

for i in range(3,a-3):
    for j in range(3,b-3):
        n=0
        count=[]
        for k in range(1,16):
            if(contiguousList[i][j][k-1] != 0 and contiguousList[i][j][k] == cc
                n += 1
            else:
                count.append(n + 1)
                n = 0
        if(len(count) == 0 and n == 15):
            fastX.append(j)
            fastY.append(i)
            continue
        if(contiguousList[i][j][0] == contiguousList[i][j][15]):
            count.append(count[0] + count[len(count)-1])
        for c in range(len(count)):
            if(count[c] > nStar):
                fastX.append(j)
                fastY.append(i)
```

In [515…
```
plt.imshow(towerImage,cmap='gray')
plt.scatter(fastX,fastY,color = 'red',s=5)
```

Out[515]: <matplotlib.collections.PathCollection at 0x7fa089629cd0>



Create Contiguous list for threshhold T = 20

In [516…
```
T = 20
a,b = towerImage.shape
contiguousList=np.zeros((a,b,16))
for i in range(3,a-3):
    for j in range(3,b-3):
        for k in range(16):
            if(neighborIntensities[i][j][k] > towerImage[i][j] + T):
                contiguousList[i][j][k] = 1
```

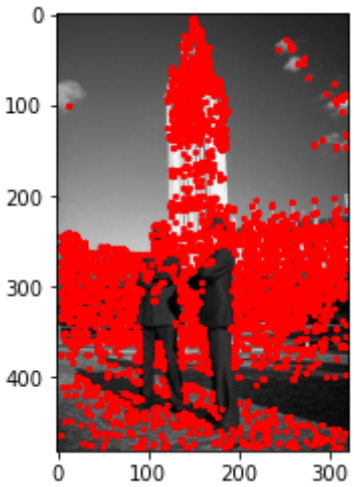
```
            elif(neighborIntensities[i][j][k] < towerImage[i][j] - T):
                contiguousList[i][j][k] = -1
```

Detect FAST feature points for threshhold T = 20

In [517…
```
fast = np.zeros((a,b))
fastX = []
fastY = []
nStar = 9

for i in range(3,a-3):
    for j in range(3,b-3):
        n=0
        count=[]
        for k in range(1,16):
            if(contiguousList[i][j][k-1] != 0 and contiguousList[i][j][k] == co
                n += 1
            else:
                count.append(n + 1)
                n = 0
        if(len(count) == 0 and n == 15):
            fastX.append(j)
            fastY.append(i)
            continue
        if(contiguousList[i][j][0] == contiguousList[i][j][15]):
            count.append(count[0] + count[len(count)-1])
        for c in range(len(count)):
            if(count[c] > nStar):
                fastX.append(j)
                fastY.append(i)
```

In [518…
```
plt.imshow(towerImage,cmap='gray')
plt.scatter(fastX,fastY,color = 'red',s=5)
```

Out[518]:
```
<matplotlib.collections.PathCollection at 0x7f9fe44d9bb0>
```



Create Contiguous list for threshhold T = 30

In [519…
```
T = 30
a,b = towerImage.shape
contiguousList=np.zeros((a,b,16))
```

```
for i in range(3,a-3):
    for j in range(3,b-3):
        for k in range(16):
            if(neighborIntensities[i][j][k] > towerImage[i][j] + T):
                contiguousList[i][j][k] = 1
            elif(neighborIntensities[i][j][k] < towerImage[i][j] - T):
                contiguousList[i][j][k] = -1
```

Detect FAST feature points for threshhold T = 30

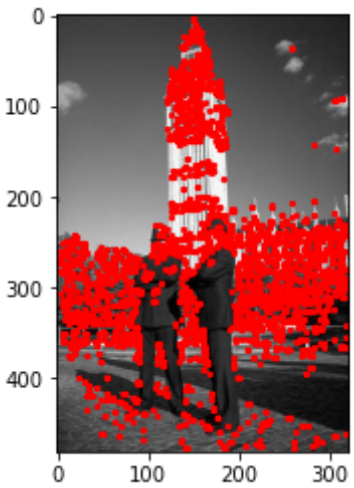
In [520…

```
fast = np.zeros((a,b))
fastX = []
fastY = []
nStar = 9

for i in range(3,a-3):
    for j in range(3,b-3):
        n=0
        count=[]
        for k in range(1,16):
            if(contiguousList[i][j][k-1] != 0 and contiguousList[i][j][k] == co
                n += 1
            else:
                count.append(n + 1)
                n = 0
        if(len(count) == 0 and n == 15):
            fastX.append(j)
            fastY.append(i)
            continue
        if(contiguousList[i][j][0] == contiguousList[i][j][15]):
            count.append(count[0] + count[len(count)-1])
        for c in range(len(count)):
            if(count[c] > nStar):
                fastX.append(j)
                fastY.append(i)
```

In [521…

```
plt.imshow(towerImage,cmap='gray')
plt.scatter(fastX,fastY,color = 'red',s=5)
```

Out[521]: <matplotlib.collections.PathCollection at 0x7fa08b48b8b0>



Create Contiguous list for threshhold T = 50

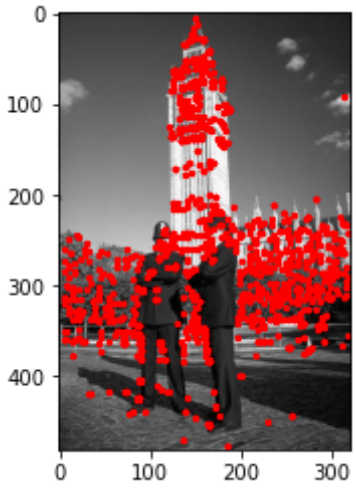
In [522…
```
T = 50
a,b = towerImage.shape
contiguousList=np.zeros((a,b,16))
for i in range(3,a-3):
    for j in range(3,b-3):
        for k in range(16):
            if(neighborIntensities[i][j][k] > towerImage[i][j] + T):
                contiguousList[i][j][k] = 1
            elif(neighborIntensities[i][j][k] < towerImage[i][j] - T):
                contiguousList[i][j][k] = -1
```

Detect FAST feature points for threshhold T = 50

In [523…
```
fast = np.zeros((a,b))
fastX = []
fastY = []
nStar = 9

for i in range(3,a-3):
    for j in range(3,b-3):
        n=0
        count=[]
        for k in range(1,16):
            if(contiguousList[i][j][k-1] != 0 and contiguousList[i][j][k] == co
                n += 1
            else:
                count.append(n + 1)
                n = 0
        if(len(count) == 0 and n == 15):
            fastX.append(j)
            fastY.append(i)
            continue
        if(contiguousList[i][j][0] == contiguousList[i][j][15]):
            count.append(count[0] + count[len(count)-1])
        for c in range(len(count)):
            if(count[c] > nStar):
                fastX.append(j)
                fastY.append(i)
```

In [524…
```
plt.imshow(towerImage,cmap='gray')
plt.scatter(fastx,fasty,color = 'red',s=5)
```

Out[524]:
```
<matplotlib.collections.PathCollection at 0x7fa01e000340>
```



Discussion

I stored intensity values of 16 possible locations located on the circular border (r = 3) of each pixel in the input image. Then for various threshholds, I generated a list of contiguous pixels, and identified which pixels can be termed as features based on nStar value (= 9). Through this, we can use FAST algorithm to detect interest points in an image. With greater threshhold values, less interest points but those carrying more information are identified.