CSC 420 A3

Yinjun Zheng

NOV, 2021

1 Part 1

1.1 Q1

1.
We need to find the ∇ that maximizes

$$convolution(\nabla^2 G(x, y, \delta), I(x, y))$$

where $I = 0$ inside the circle, $I = 1$ outside the circle

$$\begin{aligned}
\int\int \nabla^2 G(x,y,\delta) * I(x,y) d\theta dr &= \int\int_{out-circle} \frac{1}{\pi\delta^4}(\frac{r^2}{2\delta^2} - 1)e^{-\frac{r^2}{2\delta^2}} r d\theta dr \\
&= \frac{1}{\pi\delta^4}\int_r^\infty \int_0^{2\pi} \frac{r^3}{2\delta^2} e^{-\frac{r^2}{2\delta^2}} - re^{-\frac{r^2}{2\delta^2}} d\theta dr \\
&= 2\pi * \frac{1}{\pi\delta^4} * \int_r^\infty \frac{r^3}{2\delta^2} e^{-\frac{r^2}{2\delta^2}} - re^{-\frac{r^2}{2\delta^2}} dr \\
&= \frac{2}{\delta^4}[\int_r^\infty \frac{r^3}{2\delta^2} e^{-\frac{r^2}{2\delta^2}} dr + \delta^2 e^{-\frac{r^2}{2\delta^2}}|_r^\infty] \\
&= \frac{2}{\delta^4}[\int_r^\infty -\frac{1}{2} r^2 d e^{-\frac{1}{2\delta^2}r^2} + \delta^2 e^{-\frac{r^2}{2\delta^2}}|_r^\infty] \\
&= \frac{2}{\delta^4}[-\frac{1}{2} r^2 e^{-\frac{r^2}{2\delta^2}} + \int_r^\infty e^{-\frac{r^2}{2\delta^2}} d\frac{1}{2}r^2 + \delta^2 e^{-\frac{r^2}{2\delta^2}}|_r^\infty] \\
&= \frac{2}{\delta^4}[-\frac{1}{2} r^2 e^{-\frac{r^2}{2\delta^2}} - \delta^2 e^{-\frac{r^2}{2\delta^2}} + \delta^2 e^{-\frac{r^2}{2\delta^2}}|_r^\infty] \\
&= \frac{2}{\delta^4}[0 + \frac{1}{2} r^2 e^{-\frac{r^2}{2\delta^2}}] \\
&= \frac{D^2}{4\delta^4} e^{-\frac{D^2}{8\delta^2}}
\end{aligned}$$

Differentiate on δ, we get:

$$\begin{aligned}\frac{\partial}{\partial\delta}(\frac{D^2}{4\delta^4}e^{-\frac{D^2}{8\delta^2}}) &= \frac{D^2}{\delta^5}e^{-\frac{D^2}{8\delta^2}} - \frac{D^4}{16\delta^7}e^{-\frac{D^2}{8\delta^2}} = 0\\ \frac{D^2}{\delta^5} &= \frac{D^4}{16\delta^7}\\ \delta &= \frac{D}{4}\end{aligned}$$

Therefore, $\delta = \frac{D}{4}$ maximizes the magnitude of the response of the Laplacian filter.

Note that if we normalize the convolution by δ^2, then
$\int\int \delta^2\nabla^2 G(x,y,\delta) * I(x,y)d\theta dr = \int\int_{out-circle} \frac{1}{\pi\delta^4}(\frac{r^2}{2\delta^2} - 1)e^{-\frac{r^2}{2\delta^2}} r d\theta dr = \frac{D^2}{4\delta^2}e^{-\frac{D^2}{8\delta^2}}$
Then, $\delta = \frac{\sqrt{2}D}{4}$ maximizes the magnitude of the response of the Laplacian filter.
Therefore, $\delta = \frac{D}{4}$ if we do not normalize the convolution. $\delta = \frac{\sqrt{2}D}{4}$ if we normalize the convolution.

2.
The only difference from part 1 is that:
$I = 0$ outside the circle, $I = 1$ inside the circle
By the same way as part 1, we get:

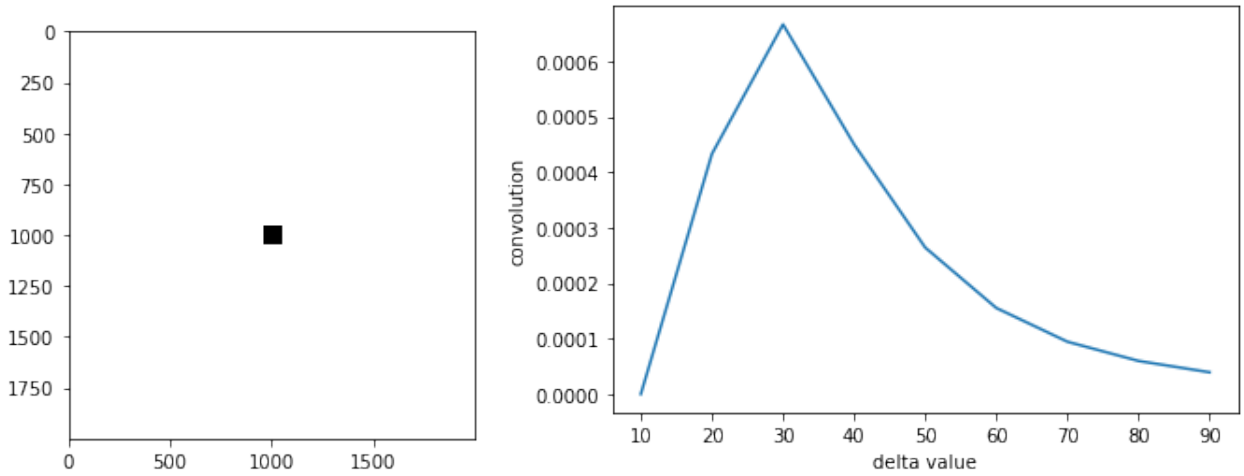
$$\begin{aligned}\int\int \nabla^2 G(x,y,\delta) * I(x,y)d\theta dr &= \int\int_{in-circle} \frac{1}{\pi\delta^4}(\frac{r^2}{2\delta^2} - 1)e^{-\frac{r^2}{2\delta^2}} r d\theta dr\\ &= \frac{2}{\delta^4}[-\frac{1}{2}r^2 e^{-\frac{r^2}{2\delta^2}} - \delta^2 e^{-\frac{r^2}{2\delta^2}} + \delta^2 e^{-\frac{r^2}{2\delta^2}}|_0^r]\\ &= \frac{2}{\delta^4}[0 - \frac{1}{2}r^2 e^{-\frac{r^2}{2\delta^2}}]\\ &= -\frac{D^2}{4\delta^4}e^{-\frac{D^2}{8\delta^2}}\end{aligned}$$

Then we get the same maximum value as in part 1.
Therefore, $\delta = \frac{D}{4}$ if we do not normalize the convolution. $\delta = \frac{\sqrt{2}D}{4}$ if we normalize the convolution.

3.
Please check the code in q1.ipynb.
I draw a 100*100 rectangle at the centre of a 2000 * 2000 platform.
I pick delta value in [0.1, 1,10,100,1000,10000,100000], then convolution value is shown as follows:

```
3.0277897366179303e-07
0.0004330503127813842
0.0006664010308770889
0.0004501258975155189
0.0002642815279600038
0.0001553910235085217
9.46126999071194e-05
5.9992913961724226e-05
3.953285118834782e-05
```

Find that the maximum value appears when δ takes value between 10 and 100. So I check the δ in [10,20,30,40,50,60,70,80,90,100], the convolution value changes as shown in the graph:



Therefore, the maximum value appears around $\delta = 30$.

1.2 Q2

1.
$N - \lambda I = \begin{pmatrix} I_x^2 - \lambda I & I_x I_y \\ I_x I_y & I_y^2 - \lambda I \end{pmatrix}$

$$\begin{aligned} det(N - \lambda I) &= (I_x^2 - \lambda I)(I_y^2 - \lambda I) - I_x I_y * I_x I_y \\ &= I_x^2 I_y^2 - \lambda I_x^2 - \lambda I_y^2 - I_x^2 I_y^2 \\ &= \lambda^2 - \lambda(I_x^2 + I_y^2) \\ &= 0 \\ \lambda_1, \lambda_2 &= 0, I_x^2 + I_y^2 \end{aligned}$$

Therefore the eigenvalues are $\lambda_1 = 0, \lambda_2 = I_x^2 + I_y^2$

2.
To prove that M is positive semi-definiteneed to show that $uMu^t >= 0 for any vector u$
Since by part 1, both eigenvalues of N is non-negative,
So N is positive semi-definite.
that is, $uNu^t >= 0$, for any vector u
Therefore, we have:

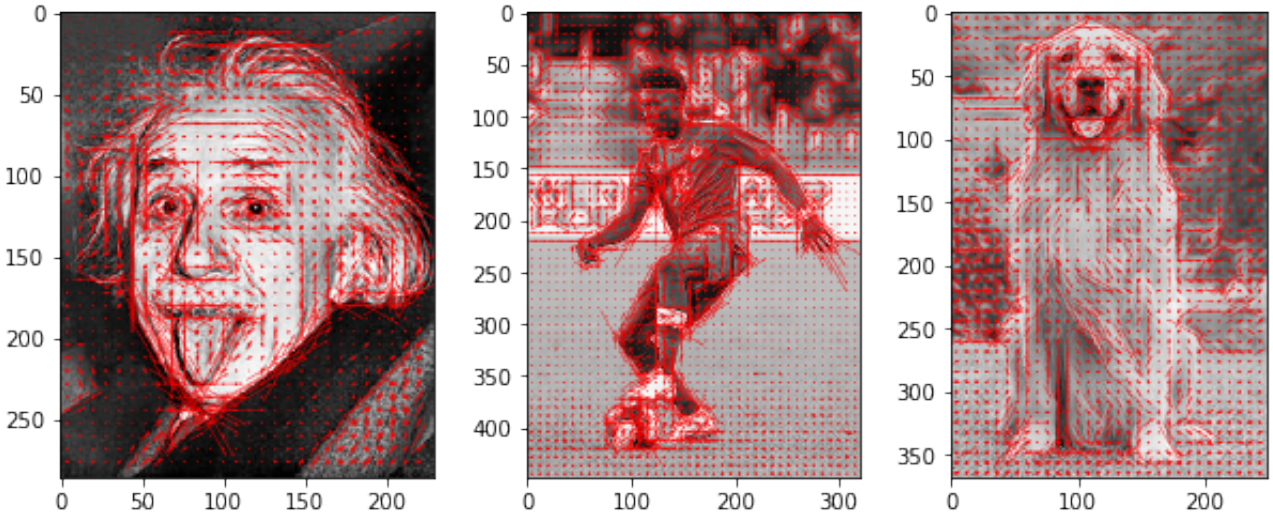
$$\begin{aligned} uMu^t &= u\sum_x\sum_y w(x,y)N(x,y)u^t \\ &= \sum_x\sum_y w(x,y)uN(x,y)u^t \end{aligned}$$

Since $w >= 0, uN(x,y)u^t >= 0$, then each term is non-negative.
So $uMu^t >= 0$ for any u.
Therefore, M is positive semi-definite.

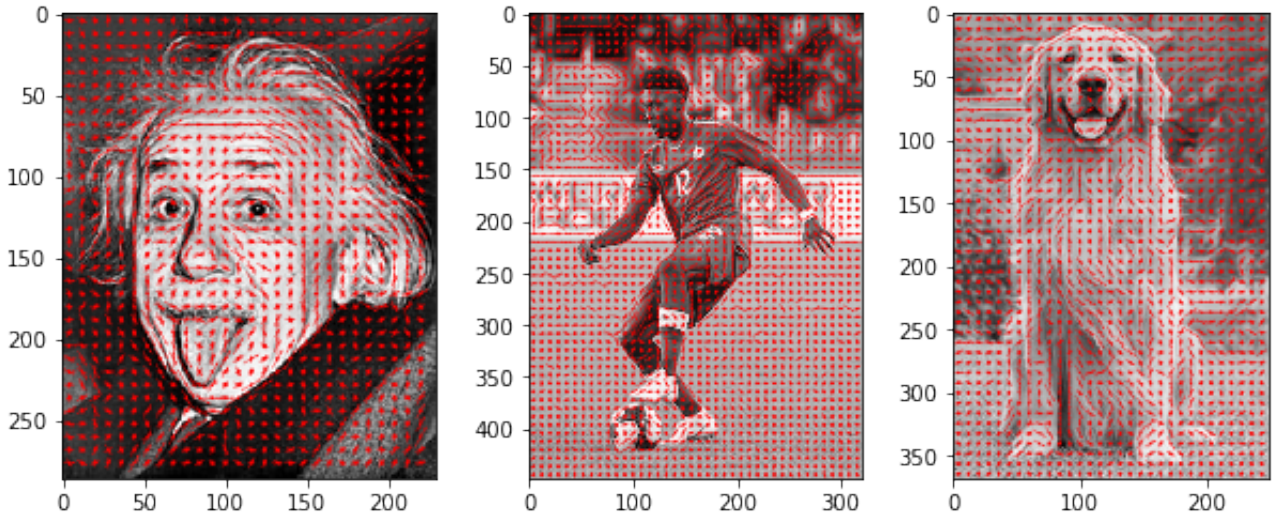
2 Part 2

2.1 Q3

Please check the detailed code in assignment3.ipynb.
1.
I choose the threshold = 10, cell size = 8
The result is shown as follows:
(1) By magnitudes



(2) By occurrence



Compare the two approaches:
The approach by magnitude has a stronger color in the main direction of the gradient, as we can see from the graph. The color is not so strong at pixels where the gradient does not change a lot. However, the approach by occurrence has a more balanced magnitude at all the pixels. Both approaches have a relative accurate description of the directions at each pixel.

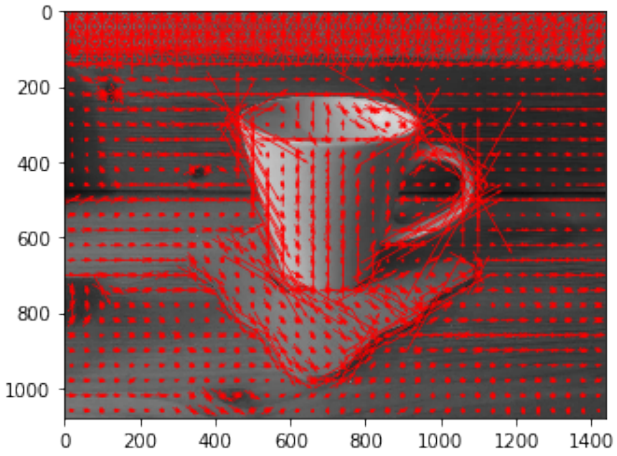
2.
The code for normalization is shown below:

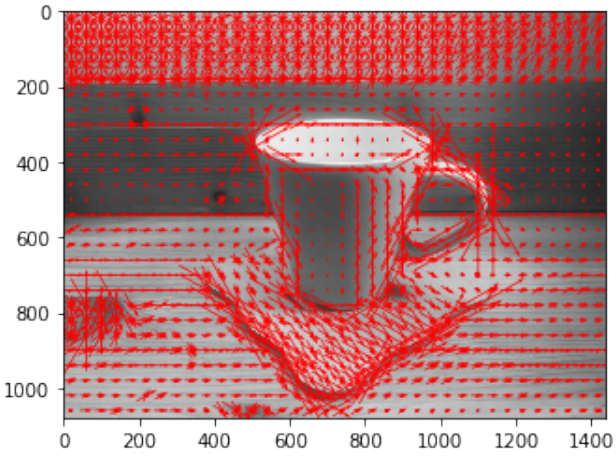
```
def Normalize(img, tao, file_path, hist, e = 0.001):
    m,n = set_cellgrid(img, tao)
    descriptor = np.zeros((m-1,n-1,24))
    for i in range(m-1):
        for j in range(n-1):
            sum_h = 0
            for k in range(6):
                sum_h += hist[i][j][k] ** 2 + hist[i+1][j][k] ** 2+ hist[i][j+1][k] ** 2 + hist[i+1][j+1][k] ** 2
                descriptor[i][j][k] = hist[i][j][k]
                descriptor[i][j][k+6] = hist[i+1][j][k]
                descriptor[i][j][k+12] = hist[i][j+1][k]
                descriptor[i][j][k+18] = hist[i+1][j+1][k]
            for k in range(24):
                descriptor[i][j][k] = descriptor[i][j][k] / math.sqrt(sum_h + e)
    write_txt(file_path, descriptor)

def write_txt(file_path, descriptor):
    f = open(file_path, "w")
    m,n,p = descriptor.shape
    for i in range(m):
        for j in range(n):
            for k in range(p):
                f.write(str(descriptor[i][j][k]) + "  ")
            f.write("\n")
    f.close()
```

Normalized histograms are saved in files ”1.txt”, ”2.txt”, ”3.txt”.
3.
I took two pictures of a cup by smart phone :

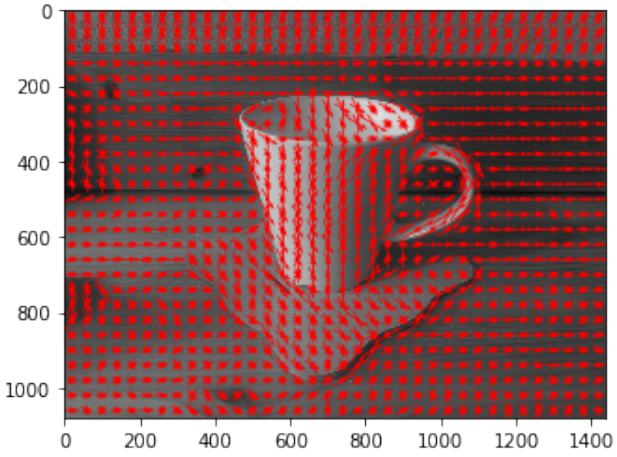

I got the HOG plot for each graph as follows:

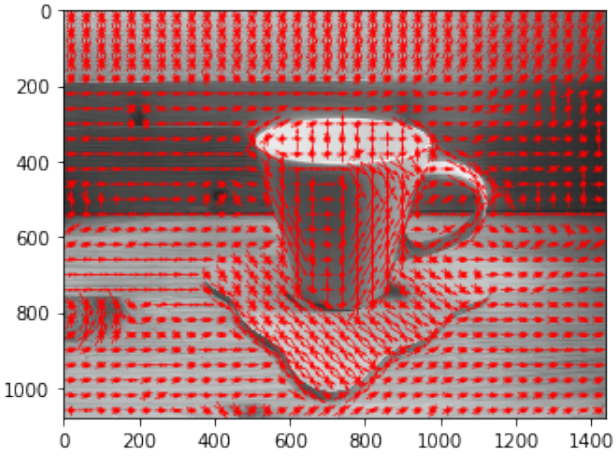
(1) By magnitude:





(2) By occurrence:





For the normalized histogram:

The Normalized vectors are stored in file ”4.txt” and ”5.txt”.

To visualize the Normalized histogram, the code is shown as follows:

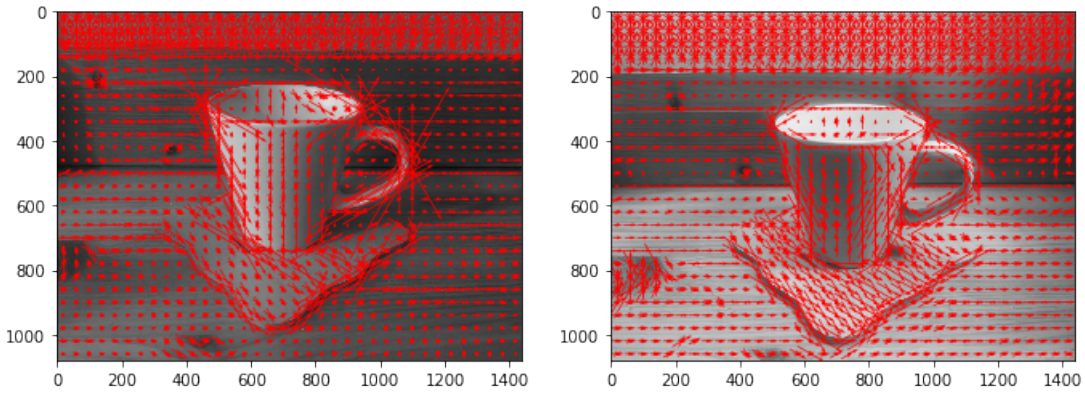
```
def transfer_Norm(hist):
    m, n= hist.shape[0]+1, hist.shape[1]+1
    hist1 = hist[:,:,:6]
    hist2 = hist[:,:,6:12]
    hist3 = hist[:,:,12:18]
    hist4 = hist[:,:,18:24]

    result = np.zeros((m,n,6))

    result[1:m-1, 1:n-1, :] = (hist1[1:, 1:,:] + hist2[:m-2, 1:, :] + hist3[1:, :n-2,:] + hist4[:m-2, :n-2,:])/4
    result[0,0,:], result[0,n-1,:], result[m-1,0,:], result[m-1,n-1,:] = hist1[0,0,:], hist2[0,n-2,:], hist3[m-2,0,:], hist4[m-2,n-2,:]
    result[0,1:n-1, :] = (hist1[0, 1:,:] + hist2[0,:n-2,:])/2
    result[m-1,1:n-1, :] = (hist3[m-2, 1:,:] + hist4[m-2,:n-2,:])/2
    result[1:m-1,0, :] = (hist1[1:, 0,:] + hist3[:m-2,0,:])/2
    result[1:m-1,n-1,:] = (hist2[1:, n-2,:] + hist4[:m-2,n-2,:])/2

    return result
```

Explanation:

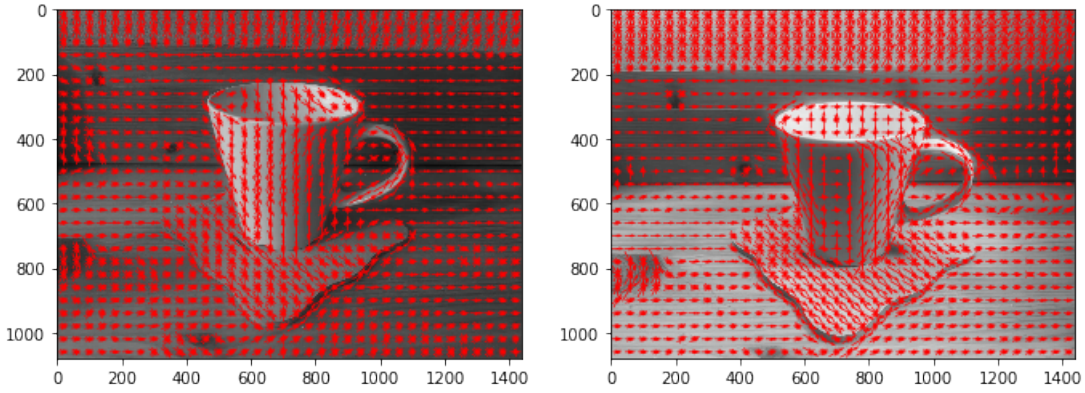
For each pixel, it has 6 directions, where each direction has 4 normalized vectors. So I add take the average of these 4 vectors for each pixel. Also, note that on the edge of the image, each pixel has 2 vectors for each direction and the corners has only one vector. Similarly, I take average of every direction. Then I get a same-shape histogram of m*n*6. i plot the graph as the same way as in the last part.

The normalized visualization graphs are shown as follows:

(1) By magnitude:



(2) By occurrence:



The left side is the result for image without flash, right side is with flash. We can see that the normalized histogram show clear directions at most of the main points, and it works well for both images with flash and without flash.

2.2 Q4

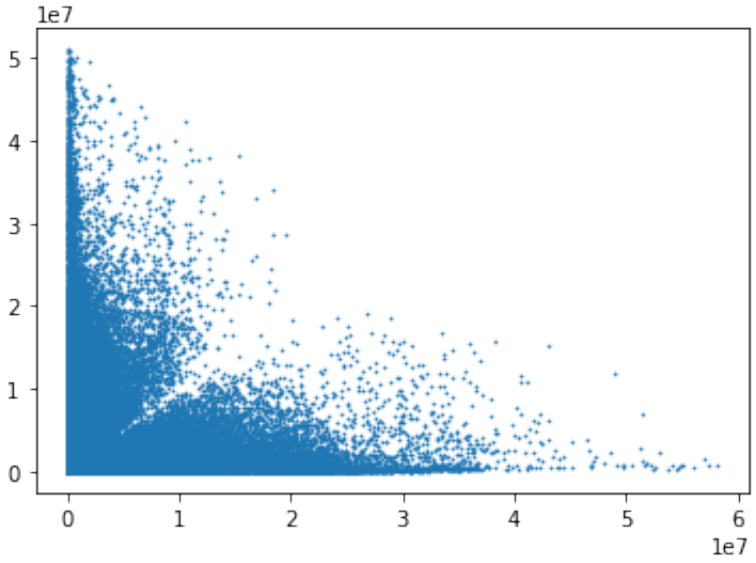
1.

Please see the code for the value of eigenvalues.

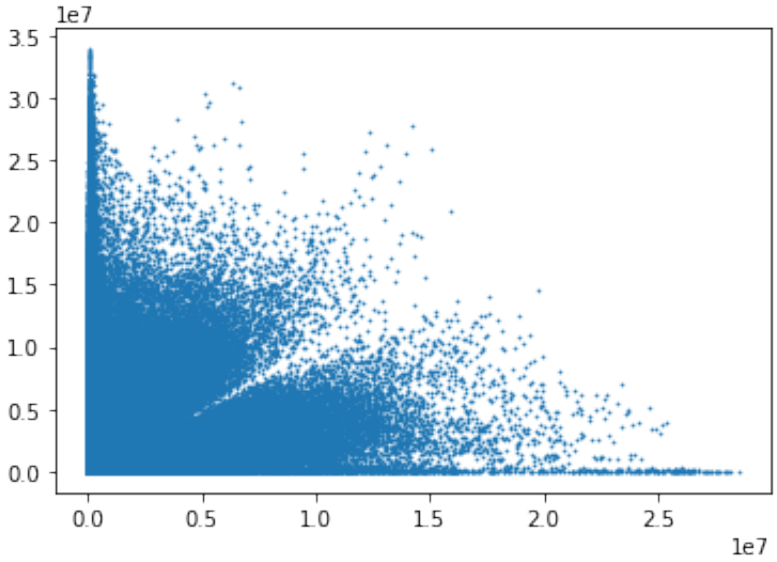
3.

The scatter plot for λ_1 and λ_2 is shown is follows:

For image 1:



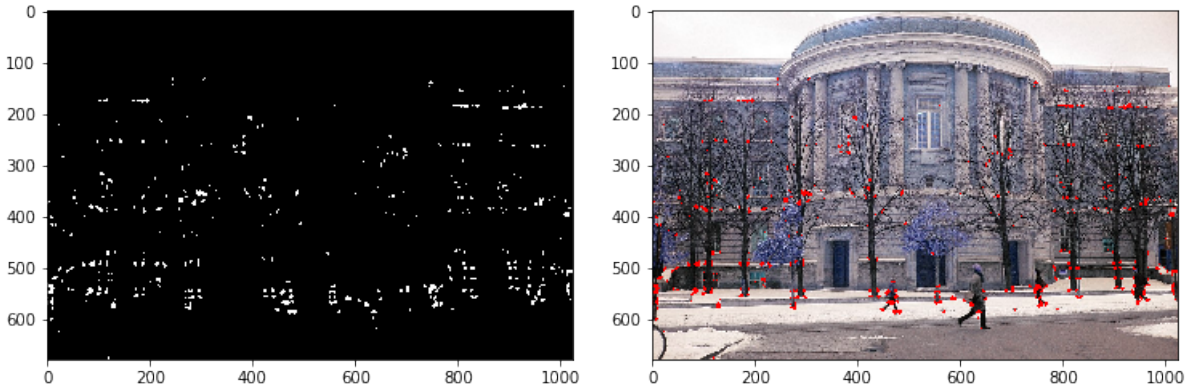
For image 2:



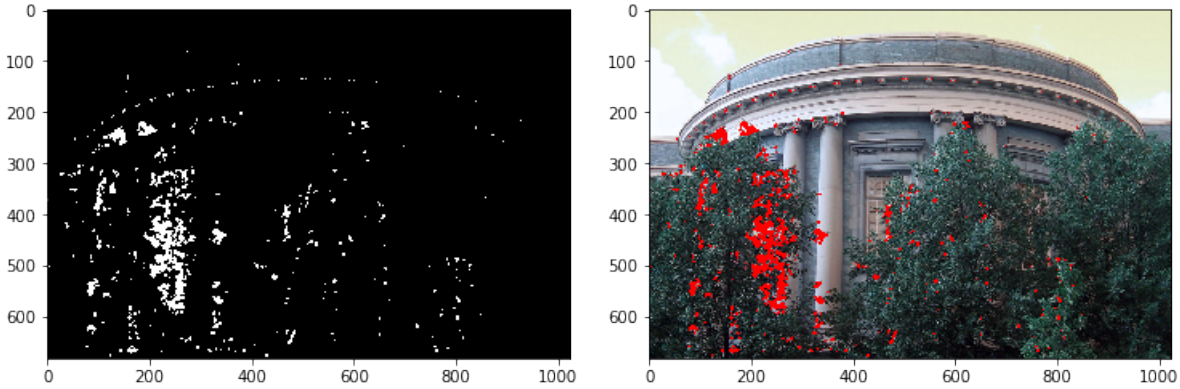
3.

I set the threshold = 0.3 * 1e7, I get the result as follows:

(1) For image 1: The red points are the corners detected.



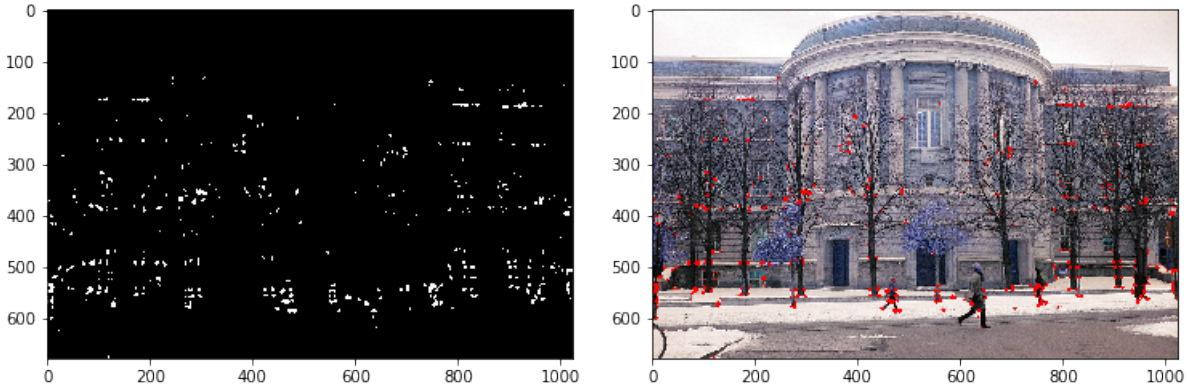
(2) For image 2: The red points are the corners detected.



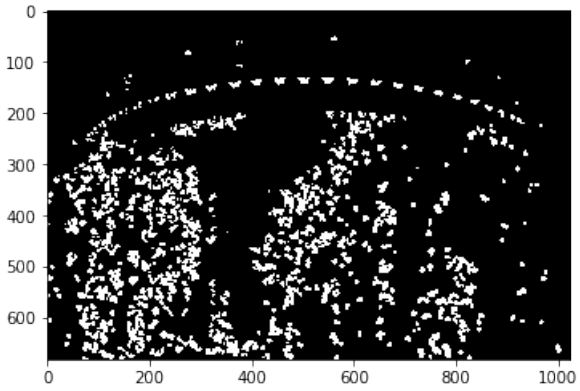
4.

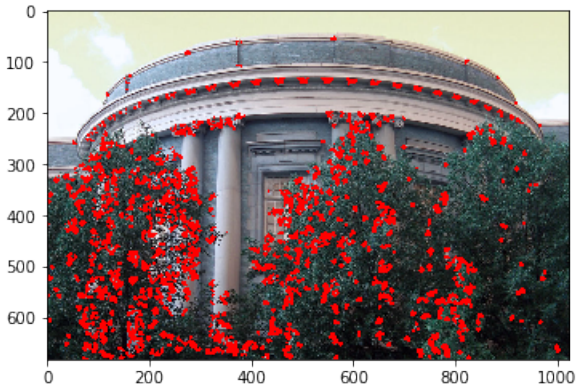
For part 3, I take $\sigma = 5$, now I change it into $\sigma = 500$. The result is shown as follows:

(1) For image 1: The red points are the corners detected.



(2) For image 2: The red points are the corners detected.





Conclusion:

When I increase the value of σ from 5 to 500, we can see from the resulting figure that the number of corner points increase. That is, if we keep the threshold unchanged, then the number of pixels with both eigenvalues greater than the threshold will increase. As we can see, with a larger σ value, we can even detect the points near the corners detected by a smaller σ value. It is probably because with higher σ value, the guassian filter has a larger value at the centre so that each pixel has a larger difference between their surroundings. o the gradient value if larger when we choose a larger σ value.