Name: Udita Gupta
Net ID: ung200
N#: N17237066

CV - Assignment 2- Practical- Report

Q1) Image Smoothing
Applied a 3*3 and 5*5 Averaging filter:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

Code and Algo Description:
There are 4 functions:
showImage(image, title): A simple matplotlib function to display the images.

padding(k, orignal_image): Used for padding an image for k*k filter size. Padding has been done with 0s on all four sides of the image. (I do have another function for padding for a m*n filter as well)

sliding_window(image, stepSize, windowSize): This function simply slides across a given image using the window_size specified. It returns a generator which is then used by the next averaging function.

averaging(k, orignal_image, padded_image, filter_mask): This function takes in the generator passed by the sliding_window function and takes only selected matrices passed to it. It will then multiply the window with the filter_mask (numpy matrix multiplication) and then sum up all the values and use that as the new value for the final image.

Name: Udita Gupta
Net ID: ung200
N#: N17237066

Output:

Original Image

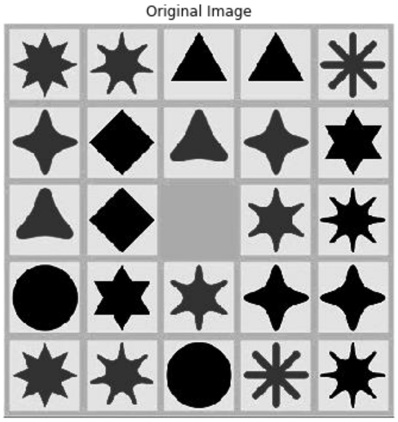


Image after 3*3 filter

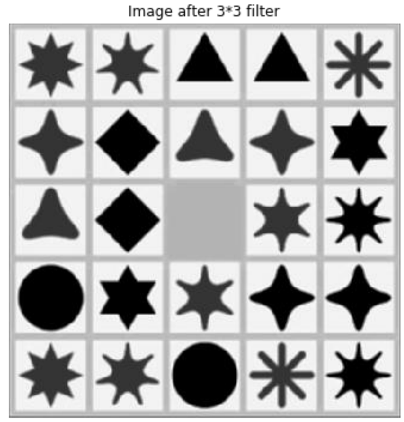
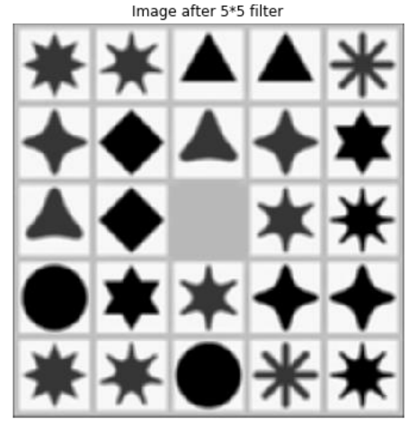


Image after 5*5 filter



Q2) Edge Detection
Functions used:
showImage(image, title)
padding(k, orignal_image)
sliding_window(image, stepSize, windowSize)
averaging(k, orignal_image, padded_image, filter_mask)

Steps:

For this we first have to find dx and dy of the original image.
I have used Sobel Operators for getting dx dy.
Dx:
Filter used:

-1	0	1
-2	0	2
-1	0	1

Dy:
Filter used:

-1	-2	-1
0	0	0
1	2	1

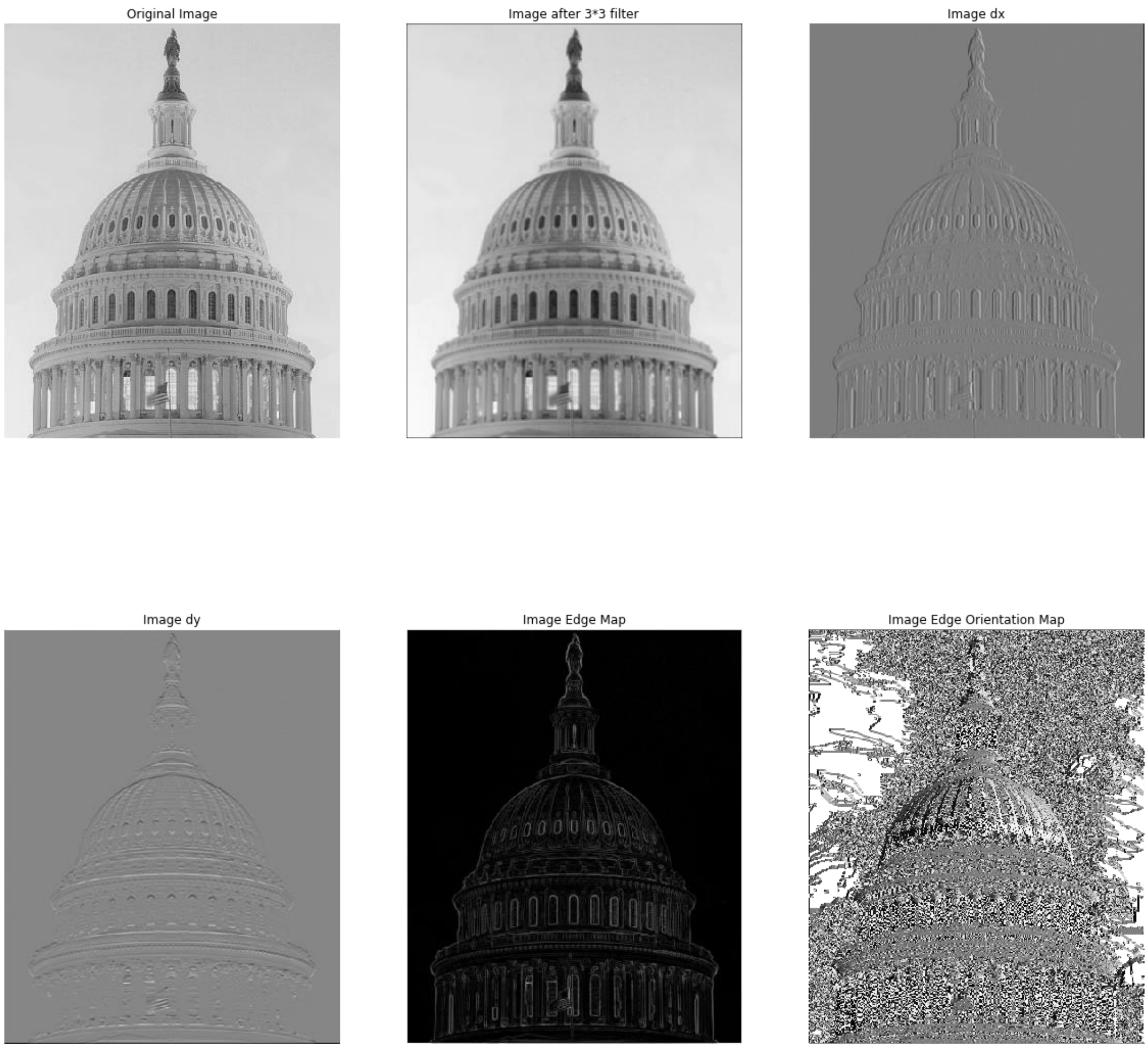
For Edge Map:
Compute the following: (We have dx and dy values already)

Name: Udita Gupta
Net ID: ung200
N#: N17237066

$$||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$$

```
For Orientation Map:
Compute the following: (We have dx and dy values)
```
$\Theta = \tan^{-1}(- (dx/dy))$

```
Output:
```



Name: Udita Gupta
Net ID: ung200
N#: N17237066

Q3) Template Matching
Functions:
zero_mean(original_matrix): This function is simply used for calculating the zero-mean matrix. That is, subtract the mean value of the matrix from each value in the matrix.

padding_not_square(m, n, orignal_image): This is same as before padding function but this time it's for a m*n filter. Padding has been done with 0s on all four sides of the image.

sliding_window(image, stepSize, windowSize): This is same as before.

averaging_not_square(m, n, orignal_image, padded_image, w_matrix): - This is the same as previous averaging function with a few new additions.
- Like mentioned in the template matching Algo in the question: the given sliding window is first re-computed using the zero-mean function.
- Then the two matrices (zero-mean window and template) are convoluted as before forming the **g** correlation matrix.
- It also computes the cross-correlation matrix **c** as mentioned in the algorithm by dividing each value of the g matrix by the magnitudes of the template matrix and window matrix.
- This function returns back both the g and c matrices.

find_threshold(matrix): This function is used for finding the threshold. We first take the c matrix returned from the above function and flatten it.
- Sort the list in ascending order
- Take the last 100 values (the max intensities)
- take an average of these 100 values as the threshold.

apply_threshold(threshold, matrix): This function takes in the threshold value and the matrix it has to apply thresholding on.
- Matrix values above threshold are kept and matrix values below the threshold are made as 0.
- We get a black image with white spots where the peaks were. Basically where we got maximum correlation between the template and the image.

Algorithm used:
- Take the input image ,i, and template, t
- Calculate zero-mean matrix of the template, lets call it zero_mean_t.
- Pad the image i

Name: Udita Gupta
Net ID: ung200
N#: N17237066

- Get all the windows (matrices) from the image i
- Calculate zero-mean of each window, lets call it w
- Convolve w and zero_mean_t, lets call it, g
- Multiply each value in matrix g by (1/(|w|| zero_mean_t |)), lets call it, c
- Get both g and c matrices and plot them
- Apply thresholding to the c matrix and get the maximum peaks.
- You can also try, direct Laplacian on matrix c.
- One more method is to smooth the matrix c and then apply laplacian to it, followed by thresholding. This basically means, LoG on matrix c and then thresholding.
- Plot all the results.

Laplacian:

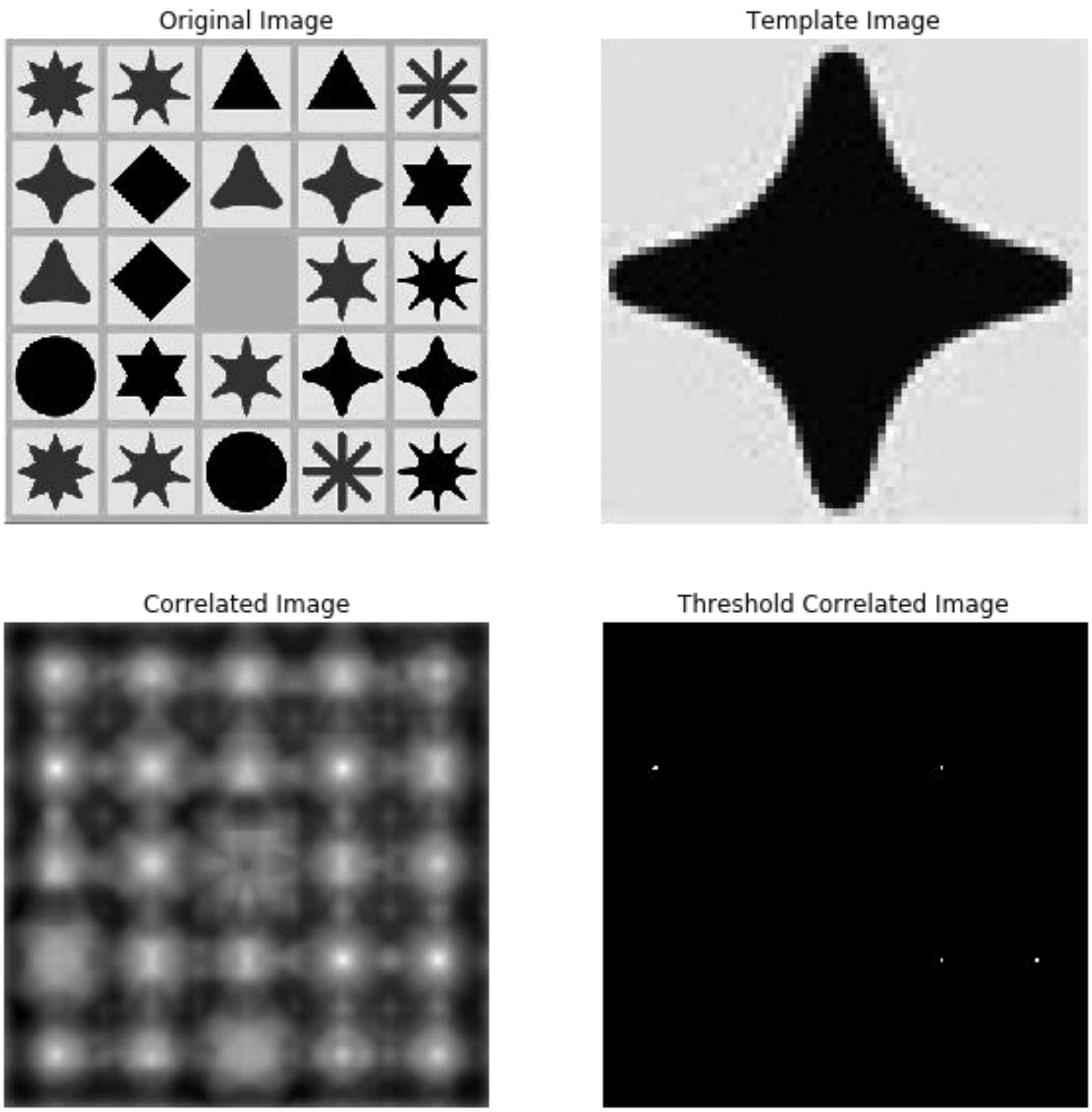
0	-1	0
-1	4	-1
0	-1	0

LoG: (Have used positive values to get bright points in the image instead of dark points)

0	0	1	0	0
0	1	2	1	0
1	2	16	2	1
0	1	2	1	0
0	0	1	0	0

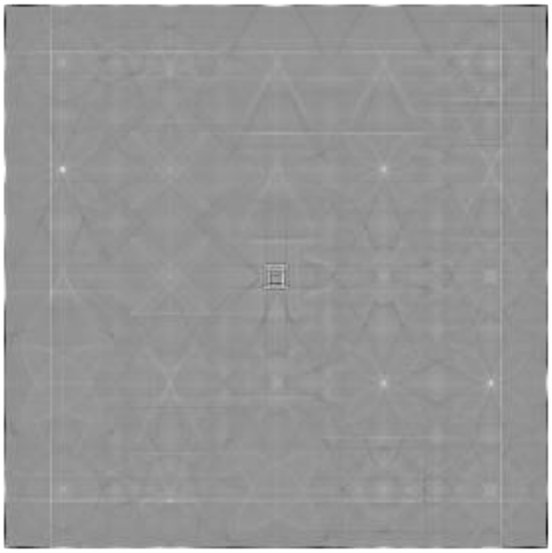
Name: Udita Gupta
Net ID: ung200
N#: N17237066

Output:

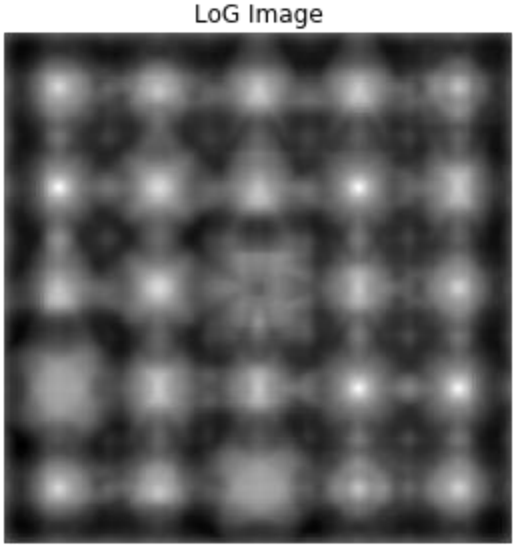


Name: Udita Gupta
Net ID: ung200
N#: N17237066

Laplacian Image



LoG Image



LoG Thresholded Image