MACHINE VISION
ASSIGNMENT 1

Name : Pavithra Ramasubramanian Ramachandran
Student ID : R00183771

Task 1 :

Code File Name: MV_Task1.py

Note:

- The code is commented with sub-task names (Task A, Task B,....Task G) and explanation of what is done in each sub-task is explained in the comments itself.
- A directory named ‘output_images_task_1’ is created in the current working directory and all output images will be stored there on execution of the code.

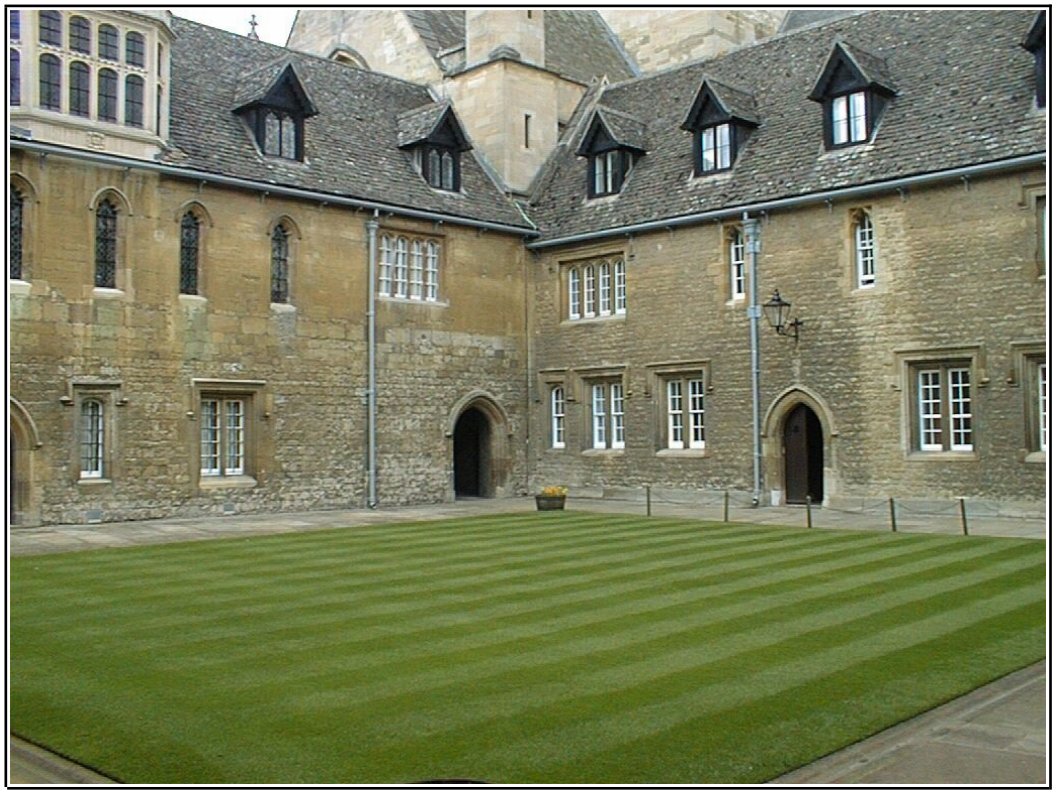
Brief Outline of approach used in Task 1:

- The input image is changed to grayscale and ‘float32’ and then resized using the resize() function in opencv.
- The scalespace representations of the resized image are generated using kernels of window size -/+ (3*sigma) and 12 images that blurr based on increasing values of sigma are displayed.
- Then 11 DoG images are calculated by finiding difference between consecutive images in the 12 scale space representations.
- The keypoints are located for every DoG image by identifying points in the image which are spatially(x,y – 8 surrounding points) and scale-wise(9 points in the scale above and 9 points in the scale below) highest.
- The gradient length, gradient direction and weighting function for every point in the 7X7 grid around the keypoint are calculated using the formulae given in the assignment pdf.
- The average oreintation angle for the keypoints are calculated using the Histograms of Orientation Gradients method where 36 bins are created representing 10 degrees of the 360 degrees each. The weighted gradient length of every point in the 7X7 grid is summed to the corresponding bin of its gradient direction. The bin with the maximum value is taken as the average orientation angle for the keypoint.
- Finally, circles with radius along the orientation angles are drawn around the keypoints with radius length 3*sigma on the original input image.

The images generated in each subtask is represented below.

Task A:

Input Image :

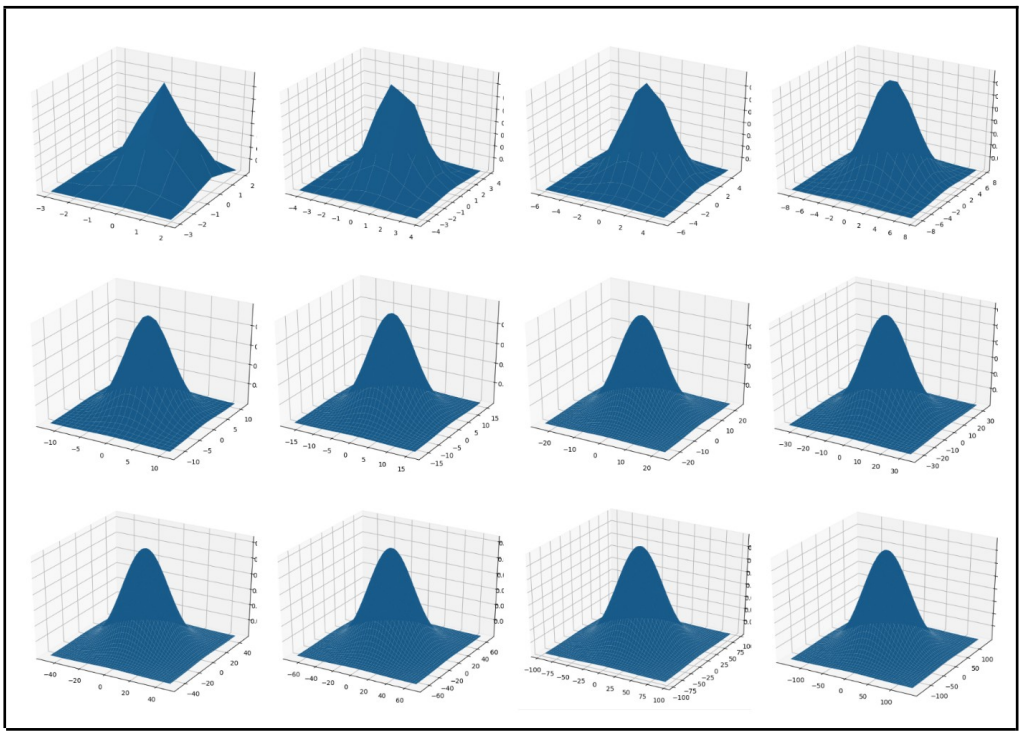


Grayscale image :

Task B:

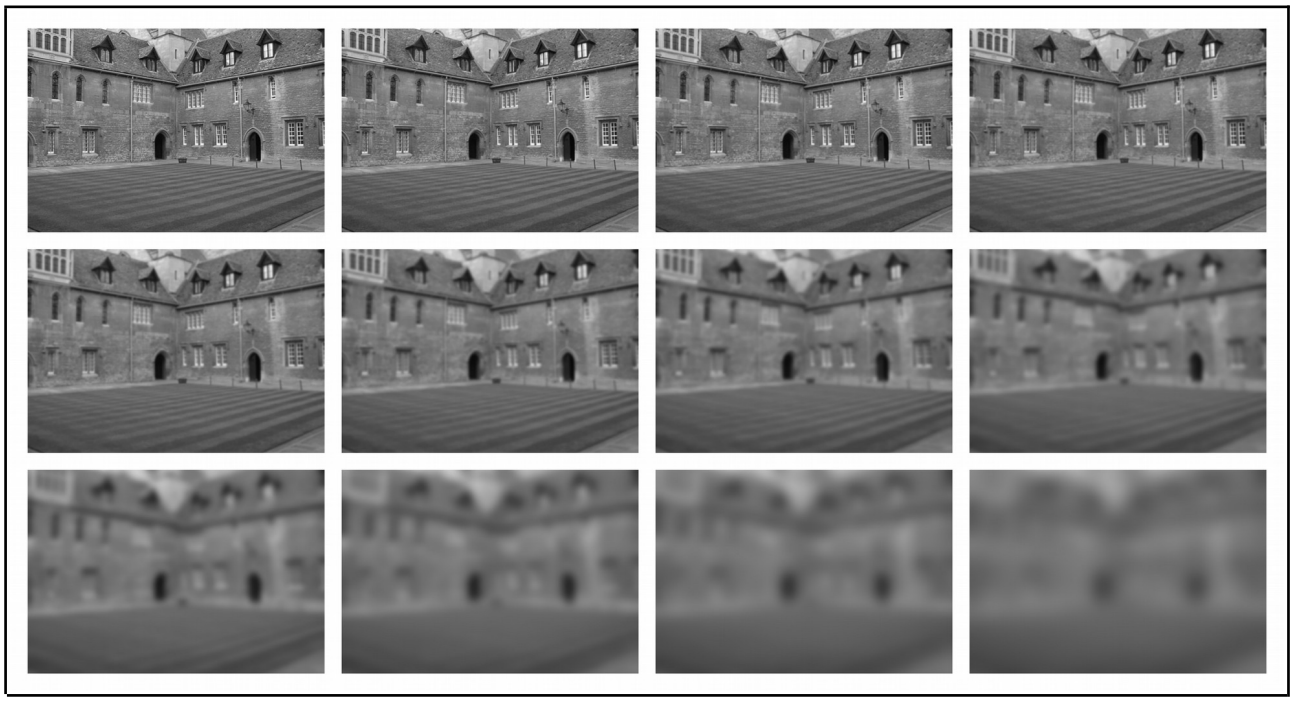
Kernel Graph Images:

The kernel graph images are represented below in order of the kernel size(left to right, top to bottom)



Guassian Smoothened Images:

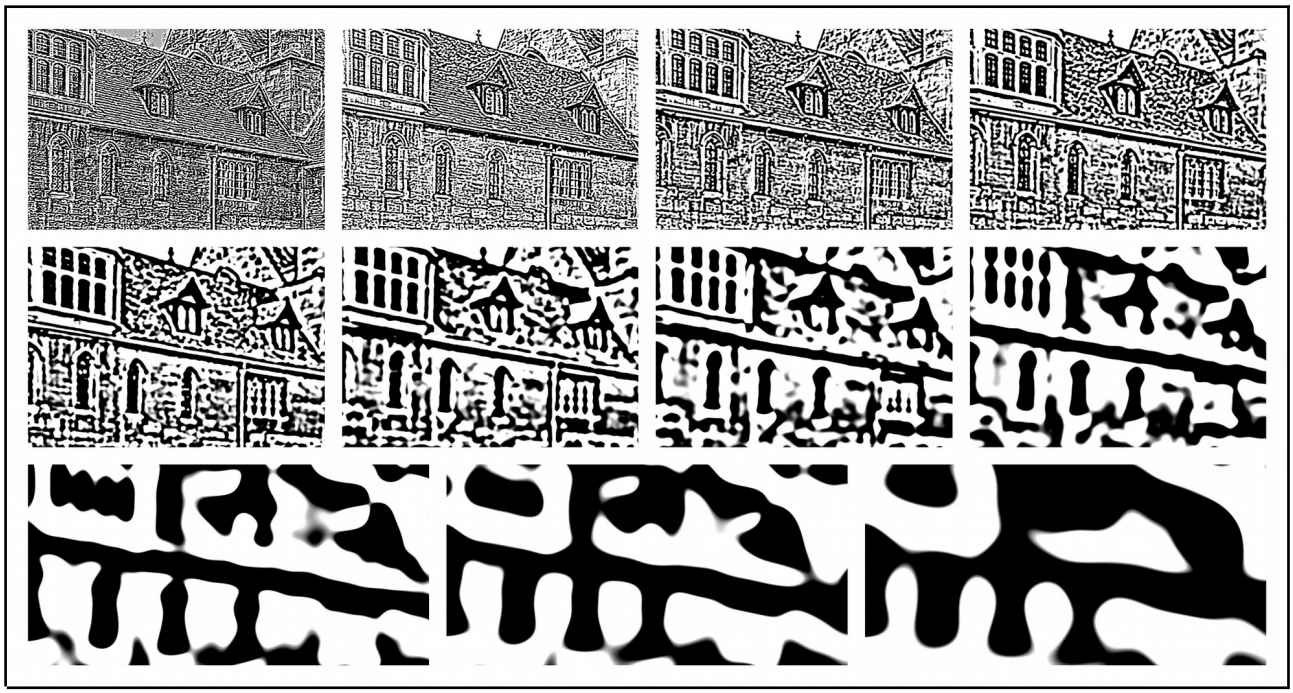
The Guassian Smoothened images are represented below with k = 0 to 11 from left to right and top to bottom in order of increasing value of k.



Task C:

DoG Images:

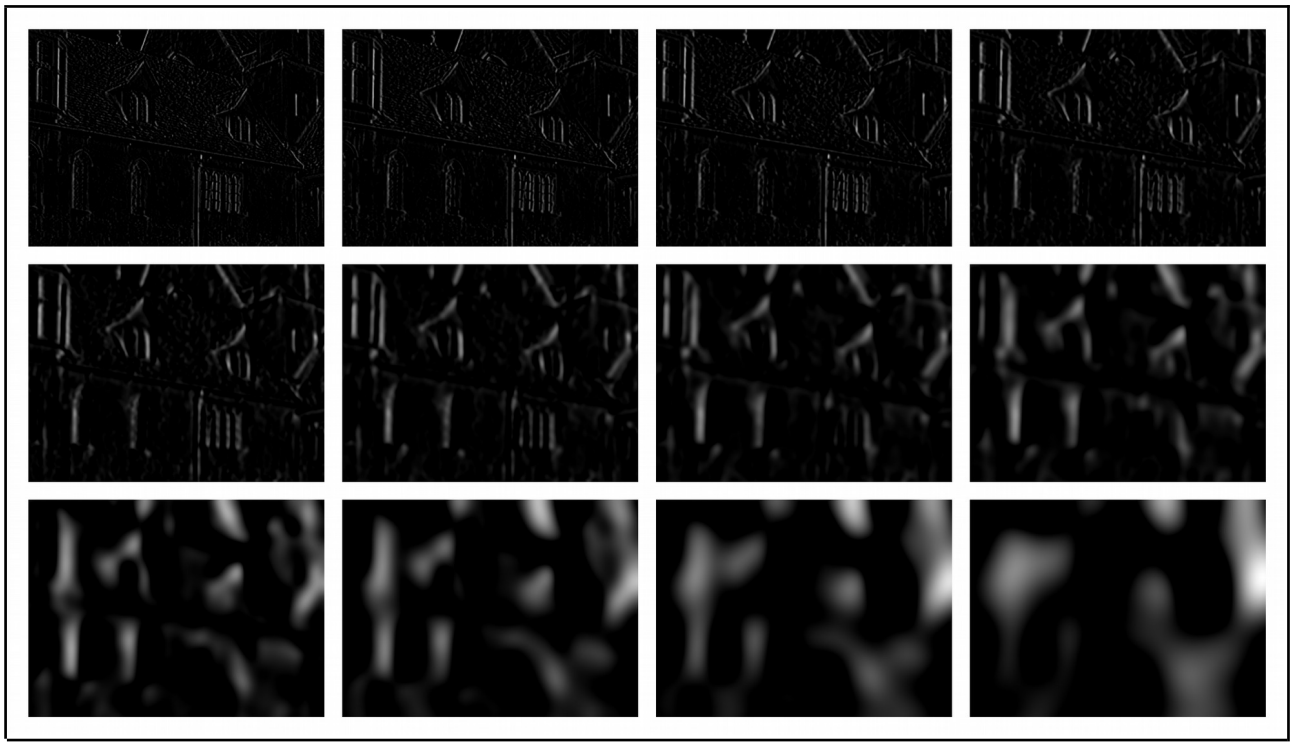
The representation below is the cropped image from the full-size images of the DoG images. All 11 images are represented below from left to right and top to bottom.



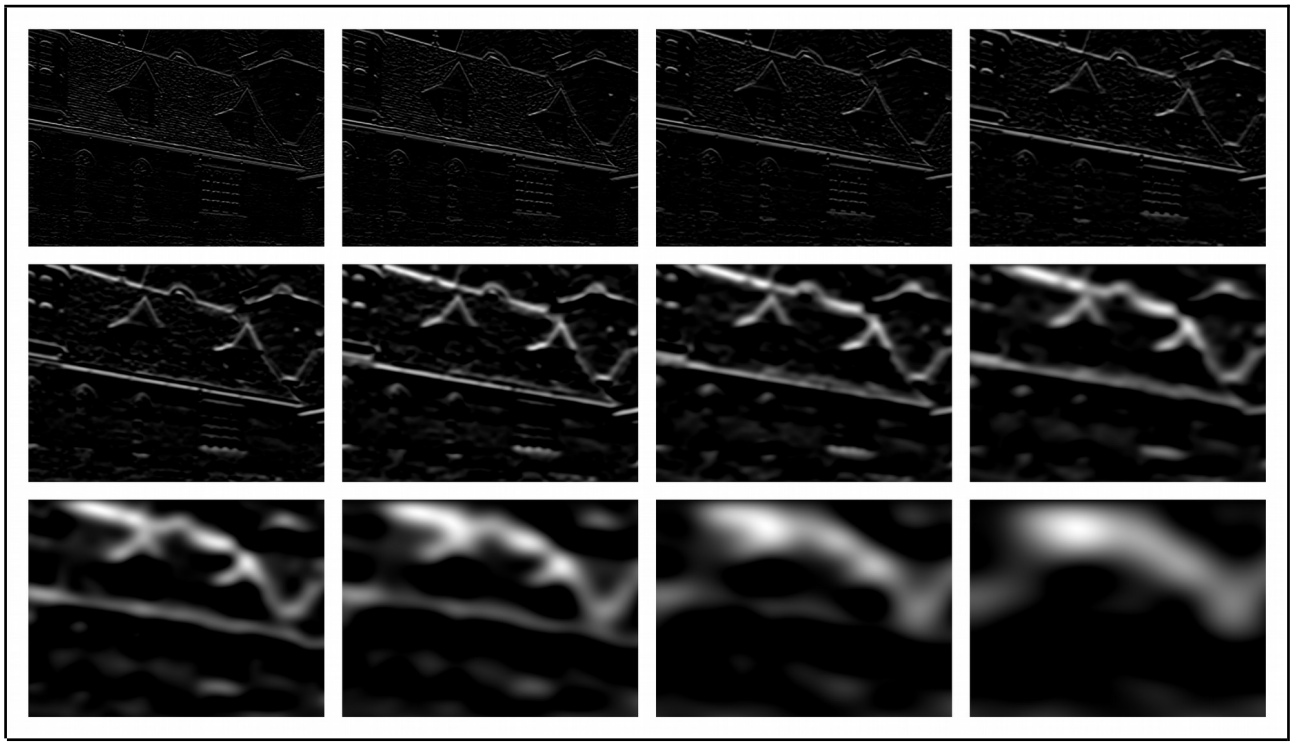
Task D :

‘X’ Derivatives of the Scale Space Representations:

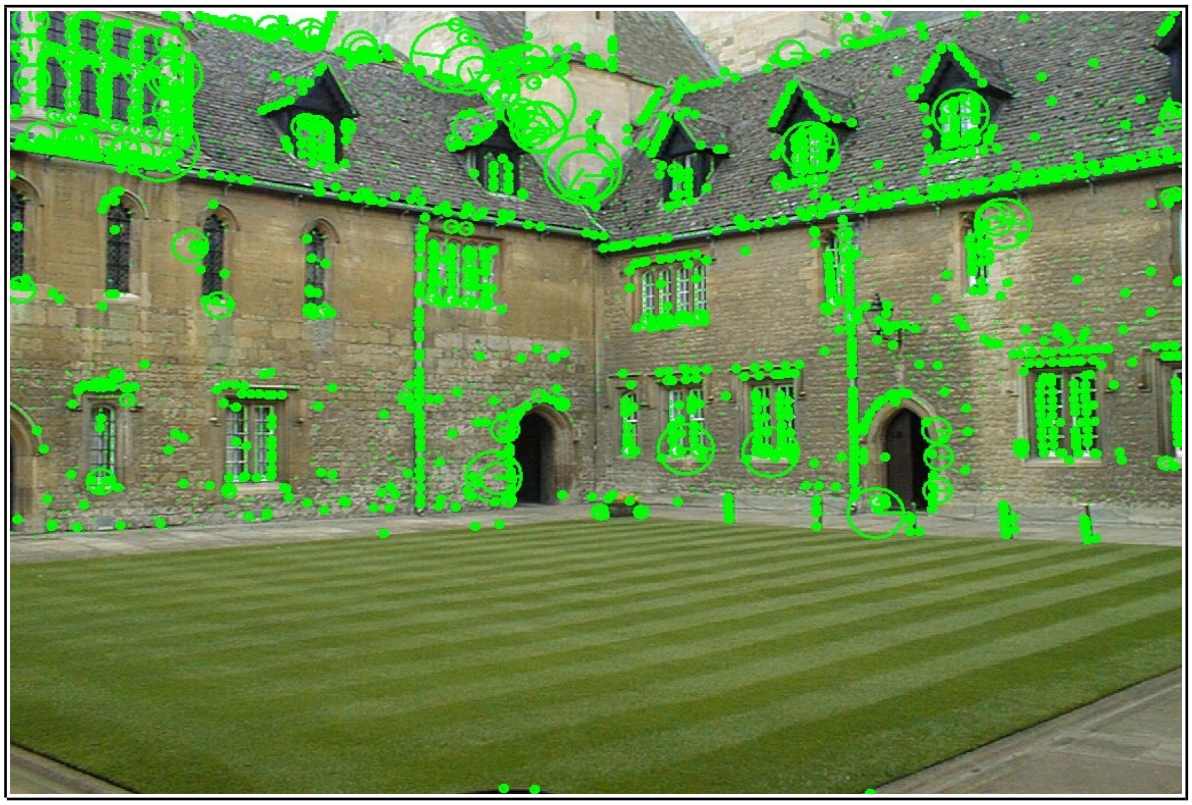
The X derivative of 12 scale space representations are represented below.



'Y' Derivatives of Scale Space Representations:
The Y derivative of 12 scale space representations are represented below.



Task G:
Final Output Image with the Keypoints:
The distinct edges in the images are identified as keypoints and circles are drawn around them.



Task 2:

Code File Name: MV_tASK2.py

Note:

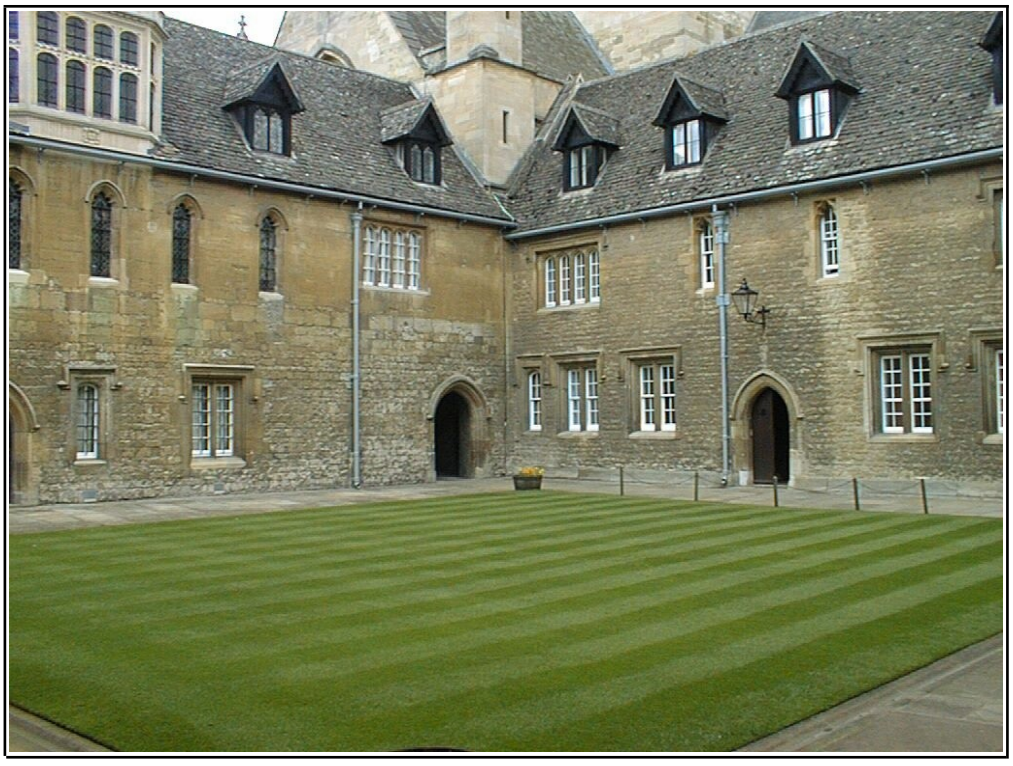
- The code is commented with sub-task names (Task A, Task B and Task C) and explanation of what is done in each sub-task is explained in the comments itself.
- A directory named ‘output_images_task_2’ is created in the current working directory and all output images will be stored there on execution of the code.

Brief Outline of approach used in Task 2:

- Both the input images are converted to grayscale and ‘float32’.
- The window is identified using the co-ordinates and using opencv a rectangle is draw around the window.
- The first image is cropped along the rectangle(slicing) and displayed. The cropped image is converted to grayscale and ‘float32’ for further processing.
- The mean and standard deviation of the cropped image is calculated.
- For every patch of the same size in the second image, the mean and standard deviation is calculated and the cross- correlation between the cropped image and the patch is calculated.
- The patch with the highest cross-correlation is identified and a rectangle is drawn around the patch in the second input image indicating the same window.

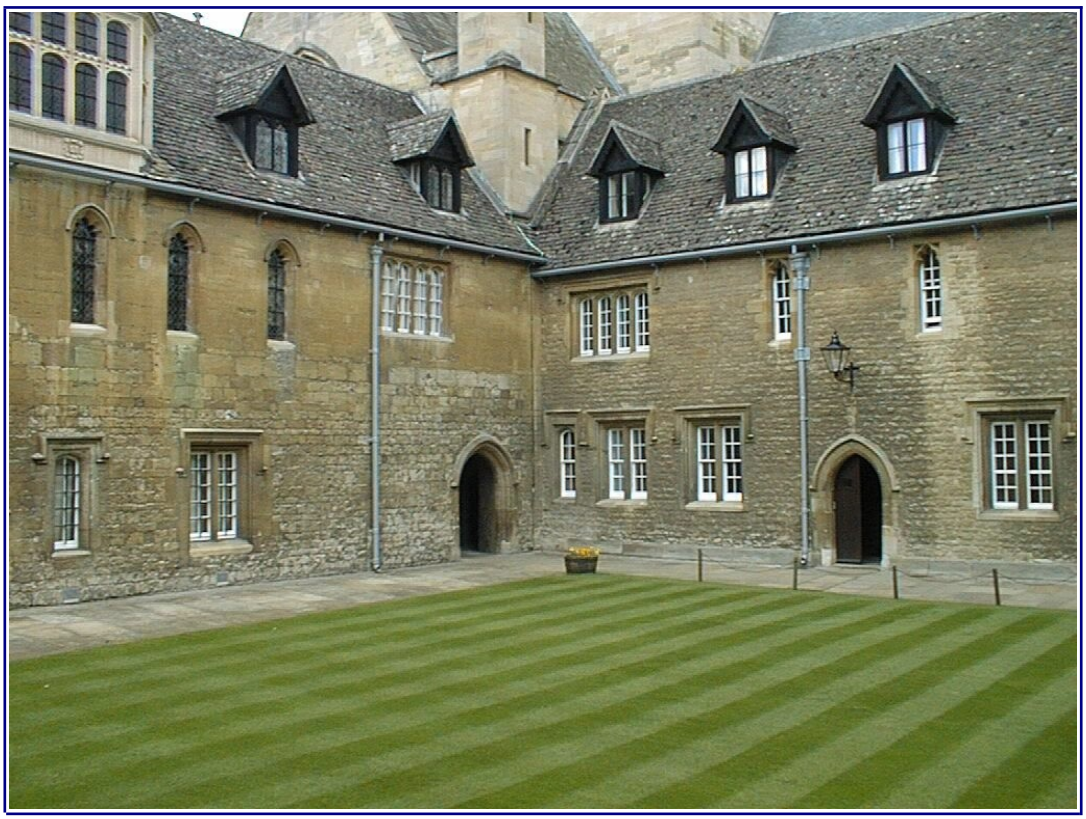
Task A:

4 images are generated:
Input Image 1:



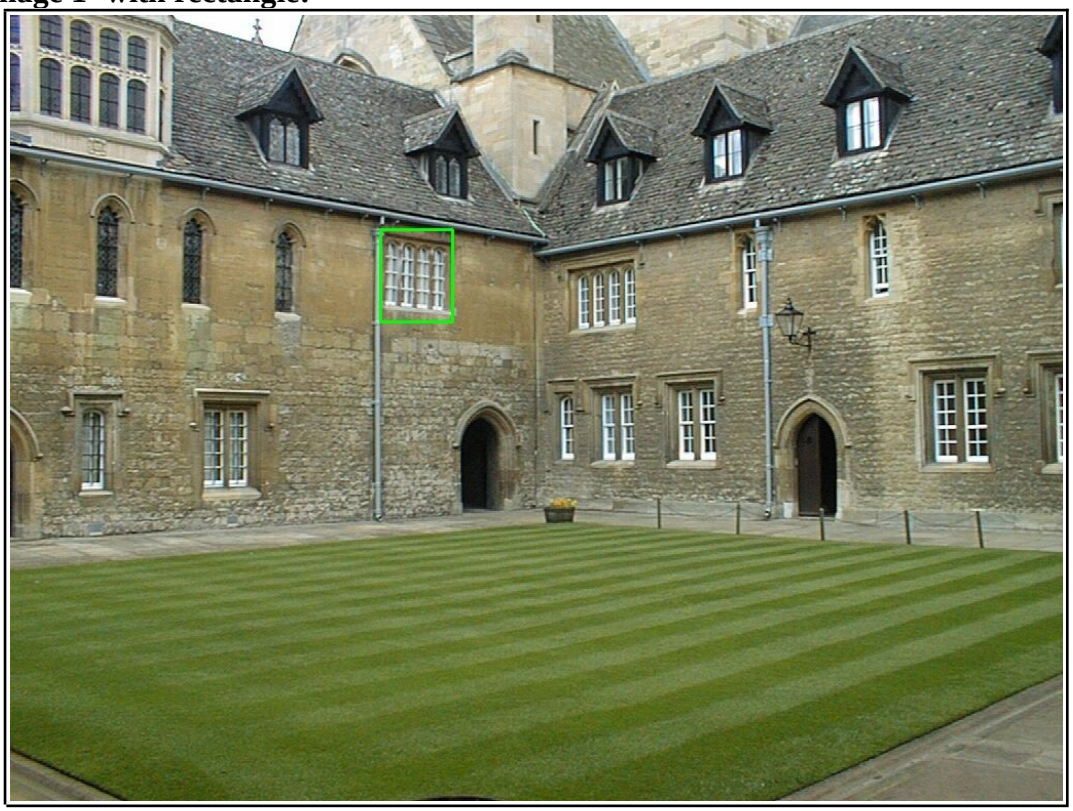
Input image 1 – grayscale:

Input Image 2:



Input Image 2 – grayscale:

Task B :
Input Image 1 -with rectangle:



Cropped Image from input image 1:

Task C :
Final Output Image – Image 2 with matching patch indicated with a rectangle drawn around: