MACHINE VISION ASSIGNMENT 1

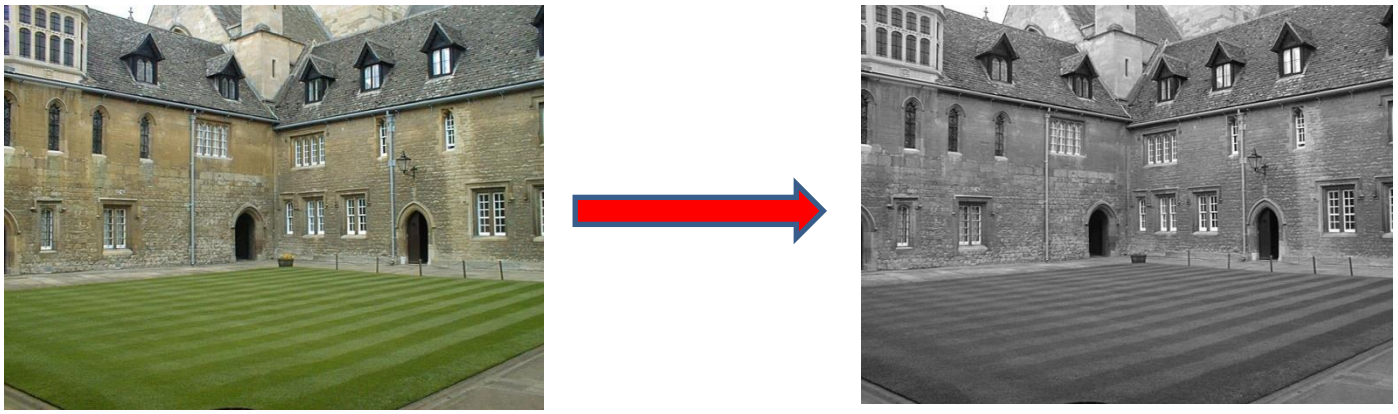
Some Points

- All the developed python code have been fully **referenced** indicating what part of the code is executing which task and the sub-parts of this assignment.
- Please go through the **ReadMe** file to understand the execution of the python file needed for the evaluation.

Task 1 – Feature Points

Part (A) - Download the input image file **Assignment_MV_01_image_1.jpg** from Canvas. Load the file and convert it into a single channel grey value image [*2 points*]. Make sure the data type is float32 to avoid any rounding errors [*1 point*]. Determine the size of the image and resize the image to double its size [*2 points*].

Solution (A) – Below we converted the input image to the single channel grey value image



We also made sure to convert the data-type of the image to float32 to avoid any rounding errors.

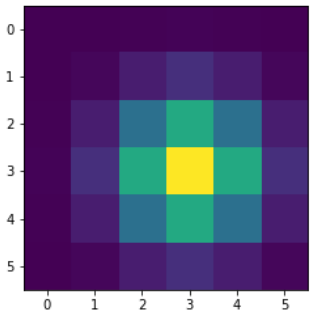
```
In [31]: runfile('C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/
Assignment 1/R00182615_MV_A01.py', wdir='C:/Users/sid/Desktop/CIT/
Semester 2/Machine Vision/Assignment 1')
Initial data type of gray value image is uint8

Final data type of the gray value image after conversion is: float32
```

Then we doubled the size of the original image as evident in the code and the snippet below as both the height and width of the original image has been doubled.

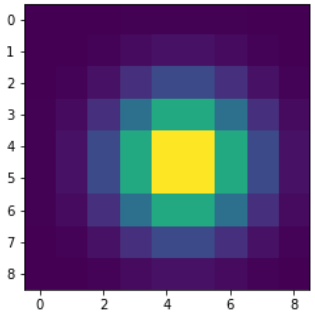
```
In [34]: runfile('C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/Assignment 1/
R00182615_MV_A01.py', wdir='C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/
Assignment 1')

Original Shape of the gray value image is (768, 1024)

Resized shape of the gray value image is (1536, 2048)
```

Part (B): Create twelve Gaussian smoothing kernels with increasing σ = 2^(k/2) , k=0,…,11, and plot each of these kernels as image [*4 points*]. Make sure that the window size is large enough to sufficiently capture the characteristic of the Gaussian. Apply these kernels to the resized input image from subtask A to create a scale-space representation and display all resulting scale-space images [*2 points*].

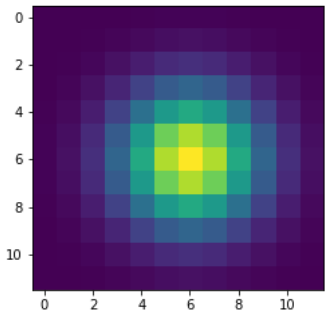
Solution (B): We have created 12 Gaussian smoothing kernels with the increasing values of sigma and are displayed below



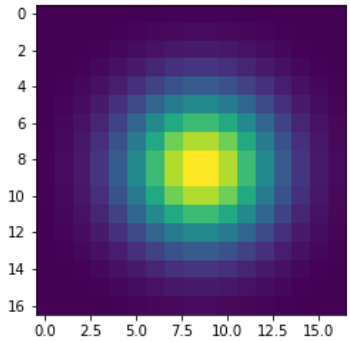
Smoothing kernel at k=0



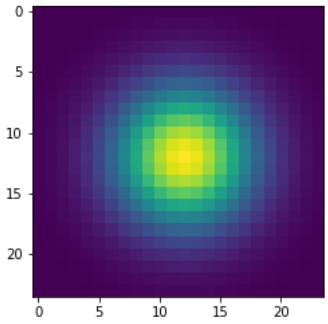
Smoothing kernel at k=1



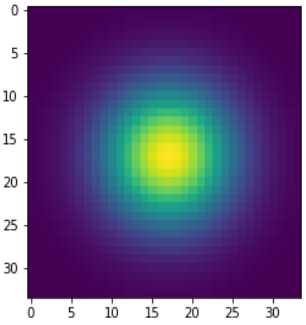
Smoothing kernel at k=2



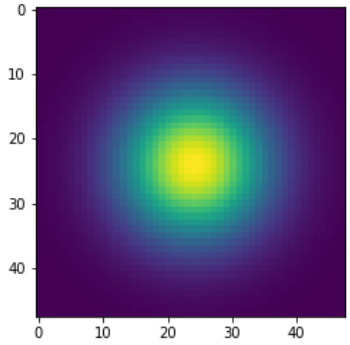
Smoothing kernel at k=3



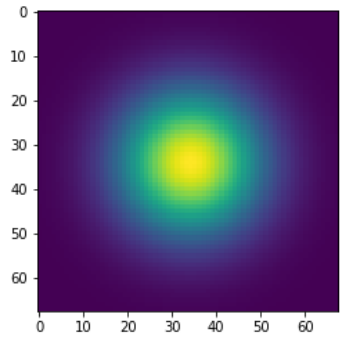
Smoothing kernel at k=4



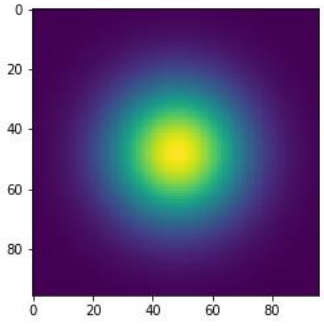
Smoothing kernel at k=5



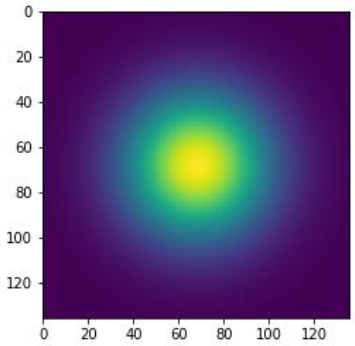
Smoothing kernel at k=6



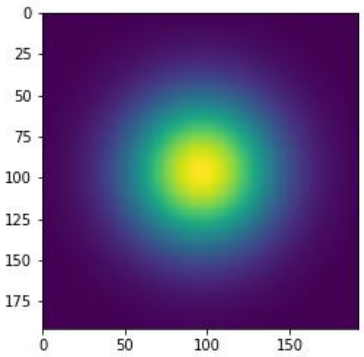
Smoothing kernel at k=7



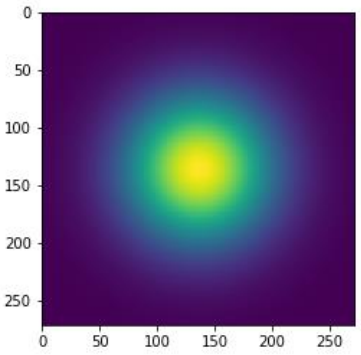
Smoothing kernel at k=8



Smoothing kernel at k=9



Smoothing kernel at k=10



Smoothing kernel at k=11

We have then obtained all the **scale space images** at respective scales from k=0 to k=11 after convoluting the input resized gray value image with these Gaussian smoothing kernels. This operation results in the blurring of the image which increases as the value of k increases.

Scale space image at k=0

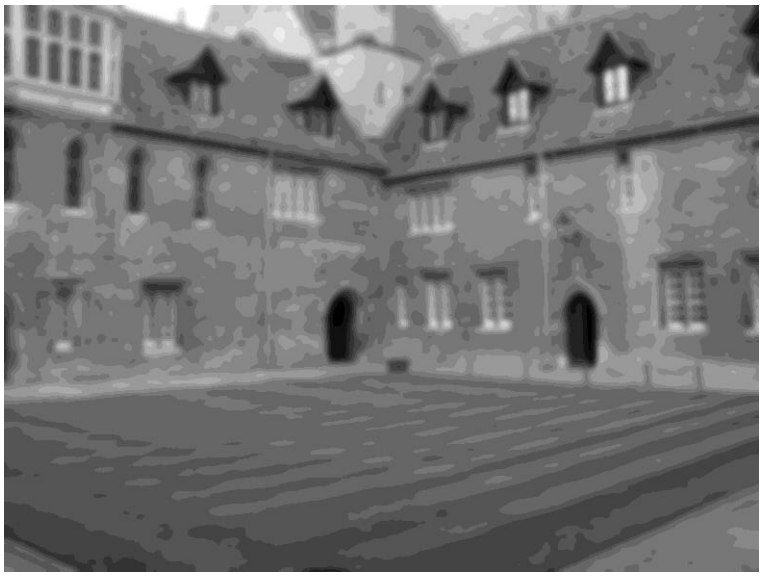
Scale space image at k=1

Scale space image at k=2

Scale space image at k=3

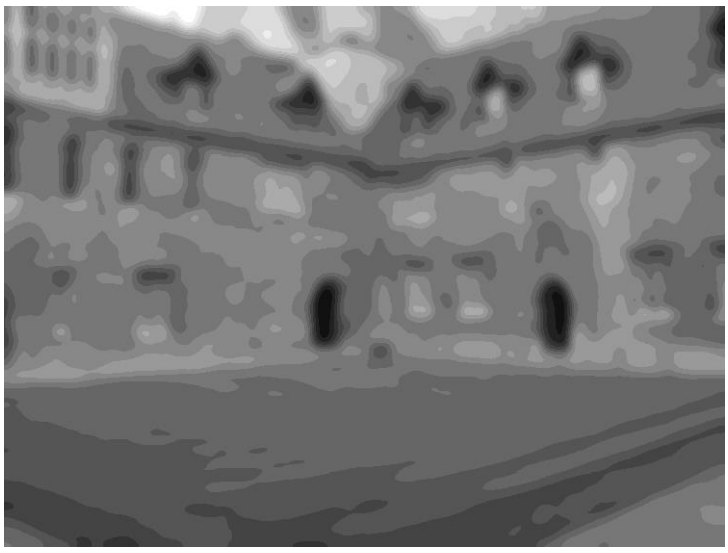
Scale space image at k=4

Scale space image at k=5

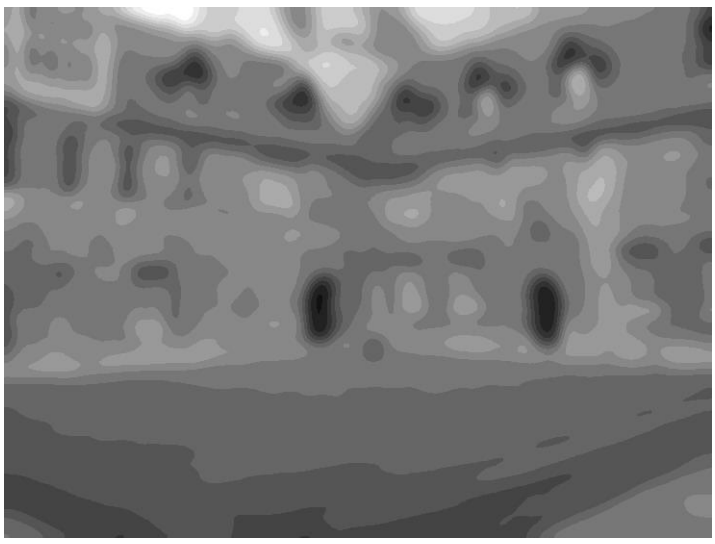


Scale space image at k=6

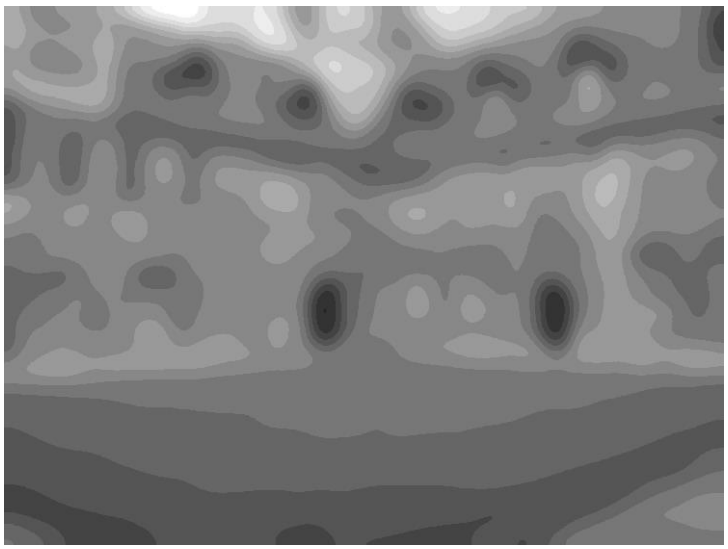
Scale space image at k=7



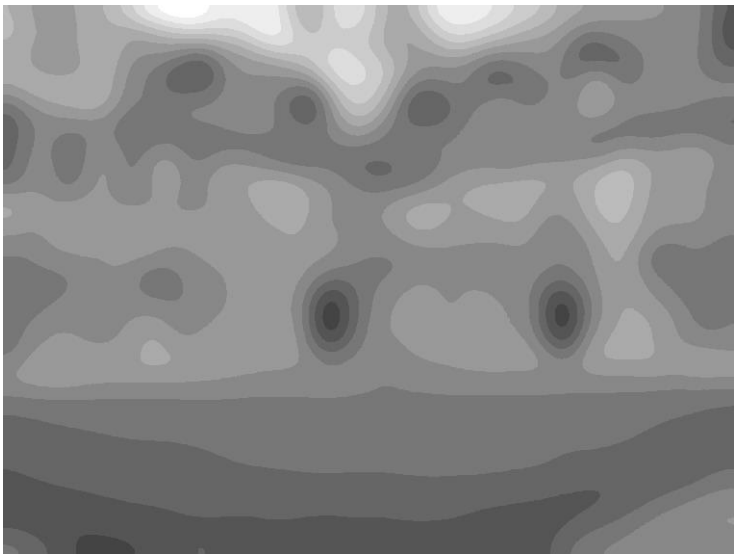
Scale space image at k=8



Scale space image at k=9



Scale space image at k=10



Scale space image at k=11

Part (C) - Use the scale-space representation from subtask B to calculate difference of Gaussian images at all scales. Display all resulting DoG images [*3 points*]

Solution (C) –

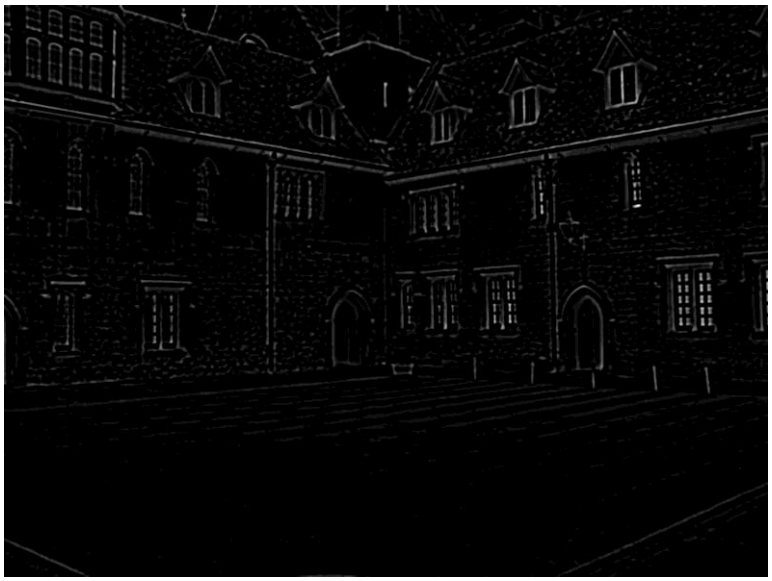
- We used all the 12 scale space images at different scales to calculate the difference of Gaussian (DoG) between the consecutive images.
- A DoG is defined as the difference between 2 adjacent parallel layer in scale space which is the discrete approximation of the Laplacian of the Gaussian which acts as a band pass filter which filters the certain frequencies in the image.
- With the DoG, we obtain the edge detectors at different scales which are the potential interest points for us.
- The result is the 11 DoG images from the 12 scale space image which are displayed below.

DoG Image 1

DoG Image 2

DoG Image 3

DoG Image 4



DoG Image 5

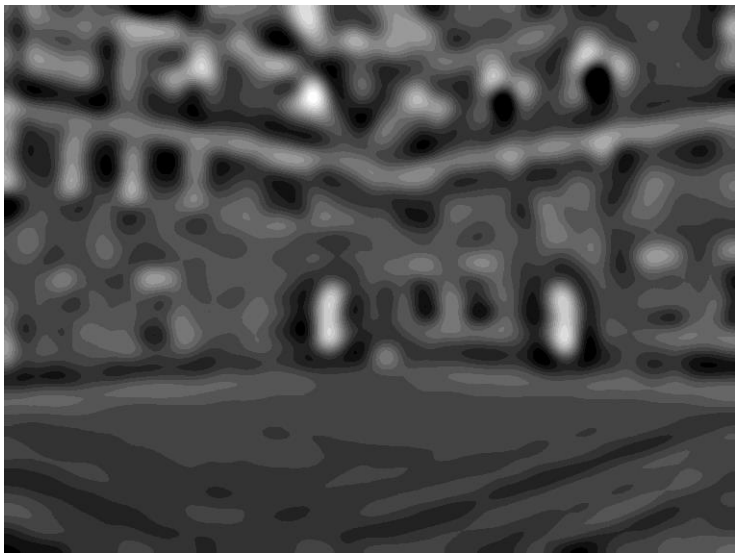
DoG Image 6

DoG Image 7

DoG Image 8

DoG Image 9

DoG Image 10

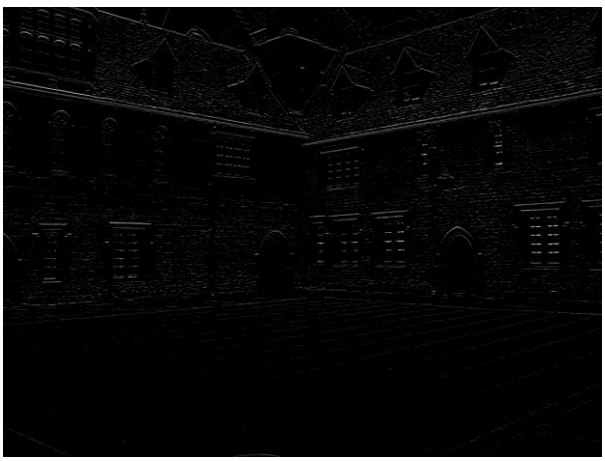


DoG Image 11

Part (E): Calculate derivatives of all scale-space images from subtask B using the kernels dx= (1 0 −1) and dy= (1 0 −1) ^ T. Display the resulting derivative images at all scales [*4 points*].

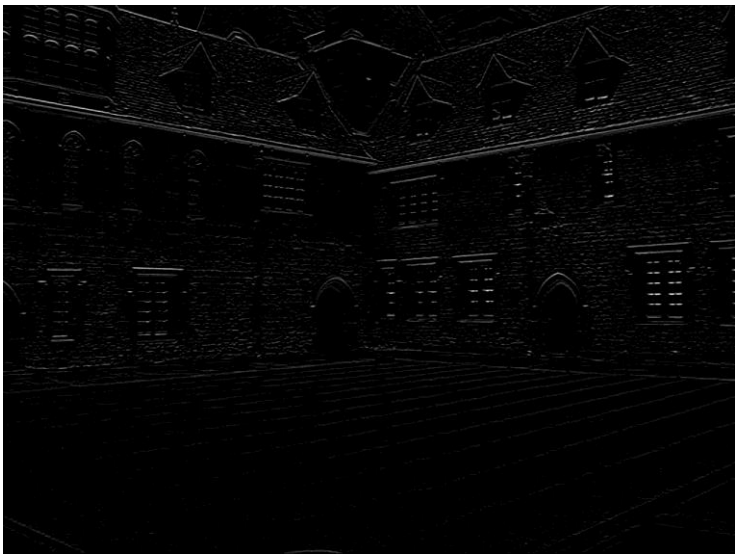
Solution (E): We have applied these kernels dx and dy to each of the scale space images. The convolution of the scale space images with these kernels results in the derivative images which signify the edge detection in x and y direction.

Derivative x image at k =0



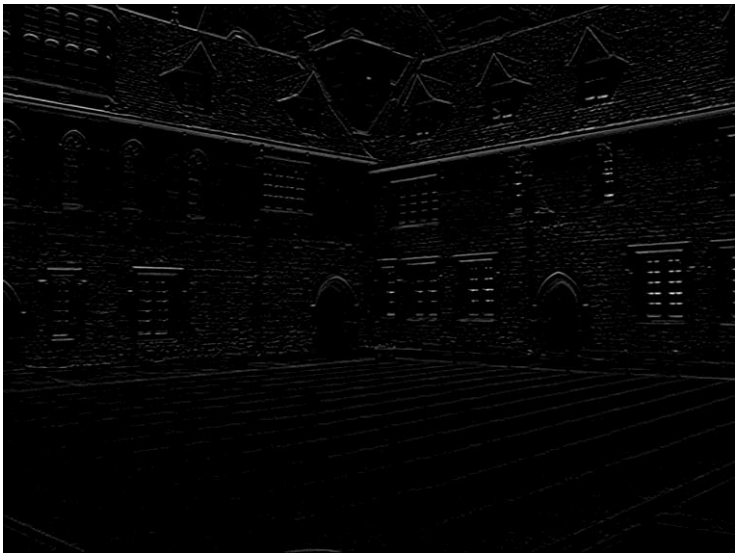
Derivative y image at k =0

Derivative x image at k =1



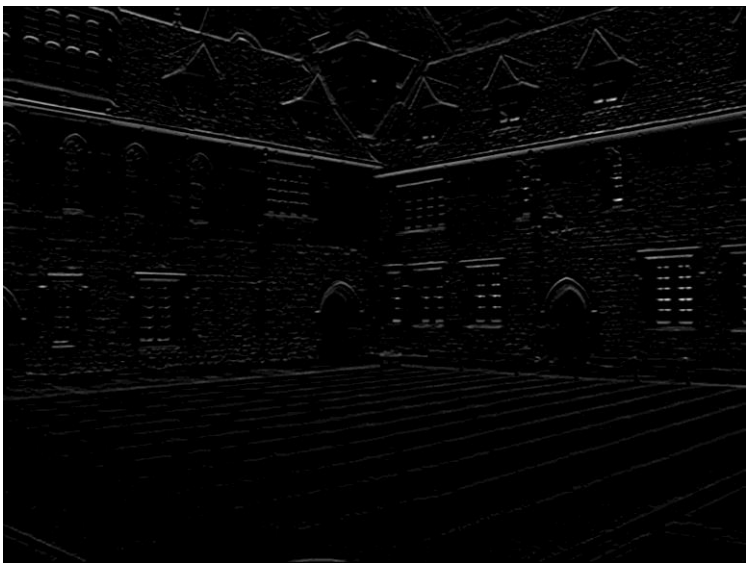
Derivative y image at k =1

Derivative x image at k =2



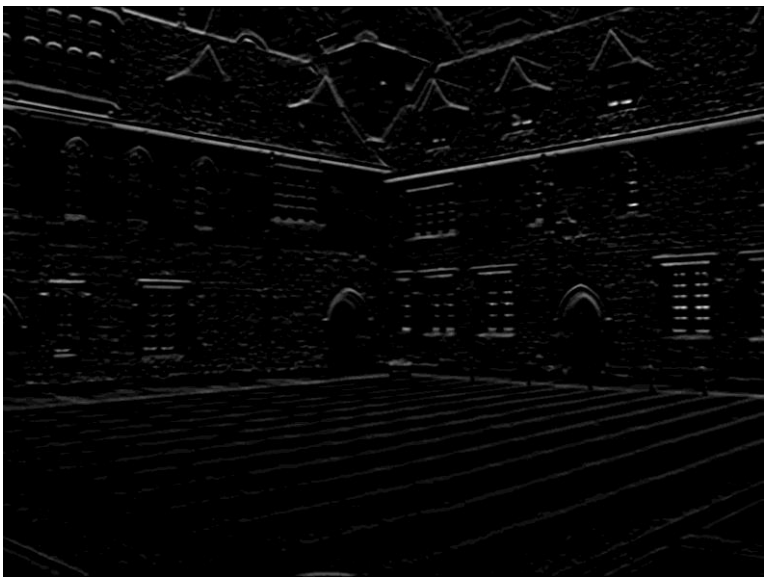
Derivative y image at k =2

Derivative x image at k =3



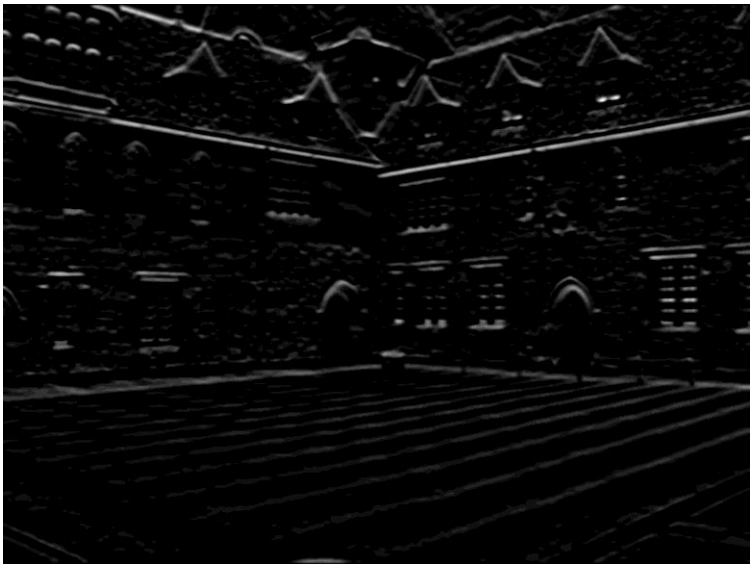
Derivative y image at k =3

Derivative x image at k =4



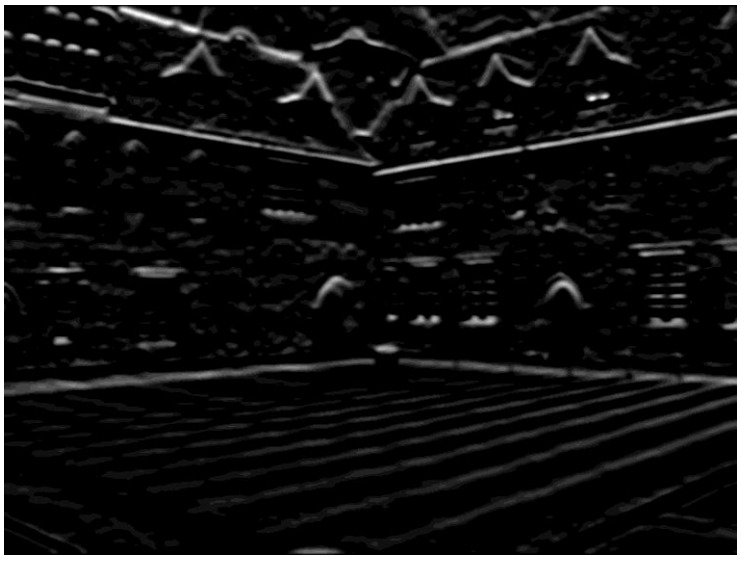
Derivative y image at k =4

Derivative x image at k =5

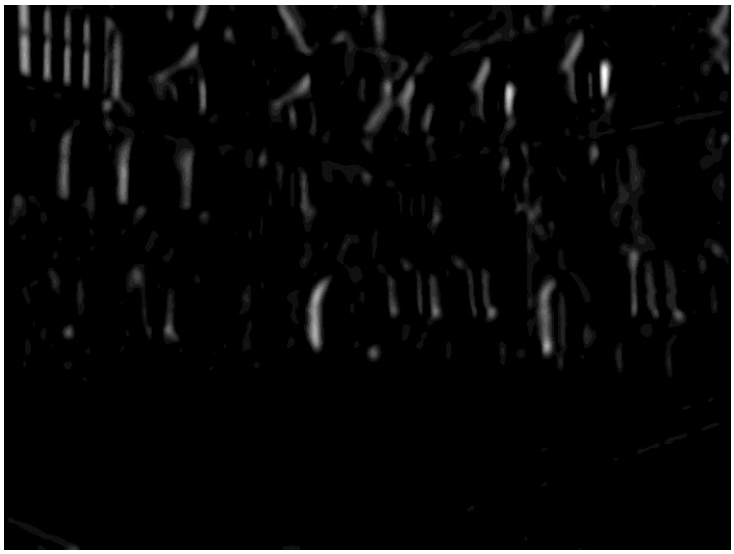


Derivative y image at k =5

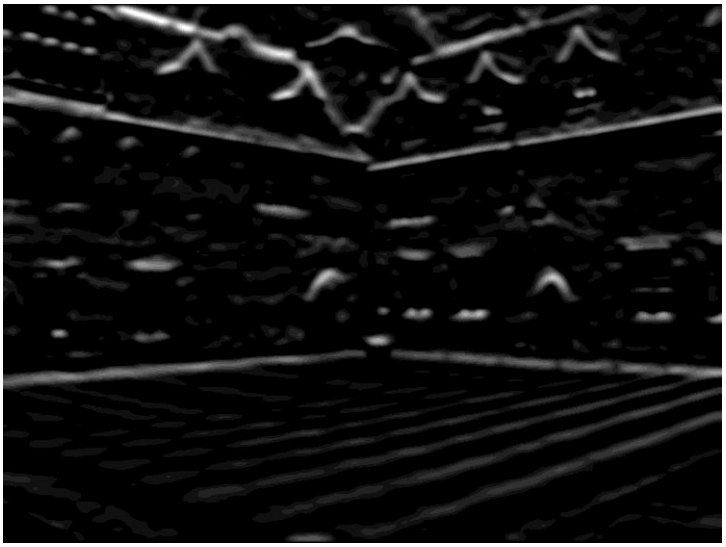
Derivative x image at k =6



Derivative y image at k =6

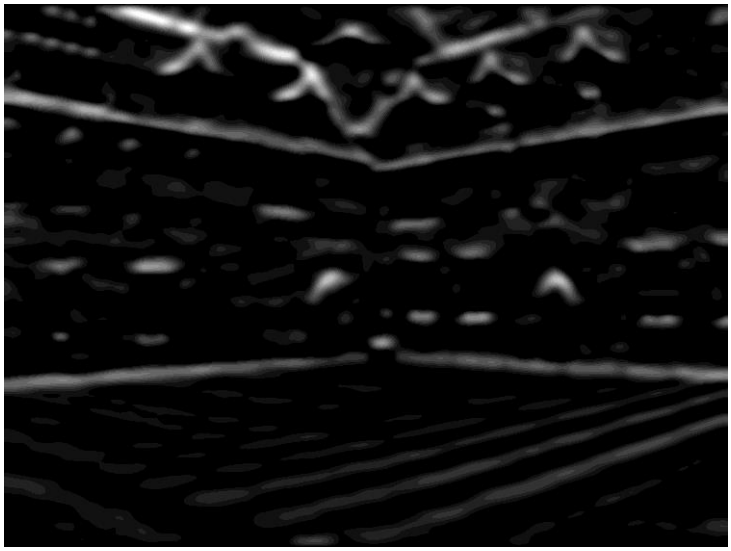


Derivative x image at k =7



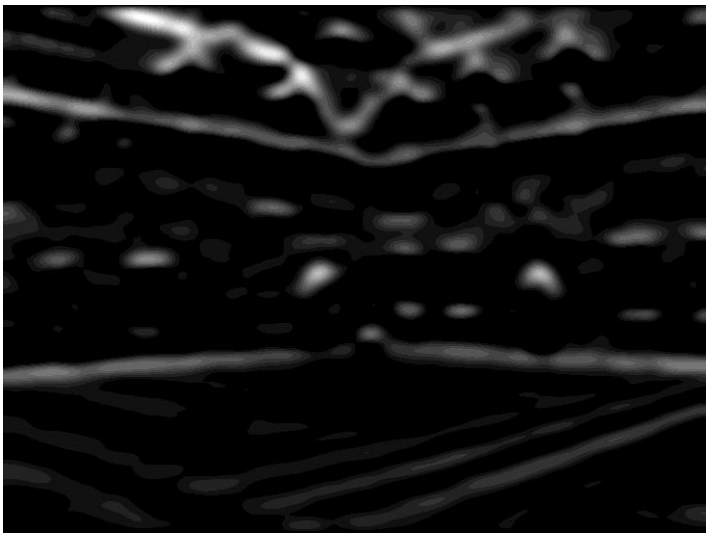
Derivative y image at k =7

Derivative x image at k =8



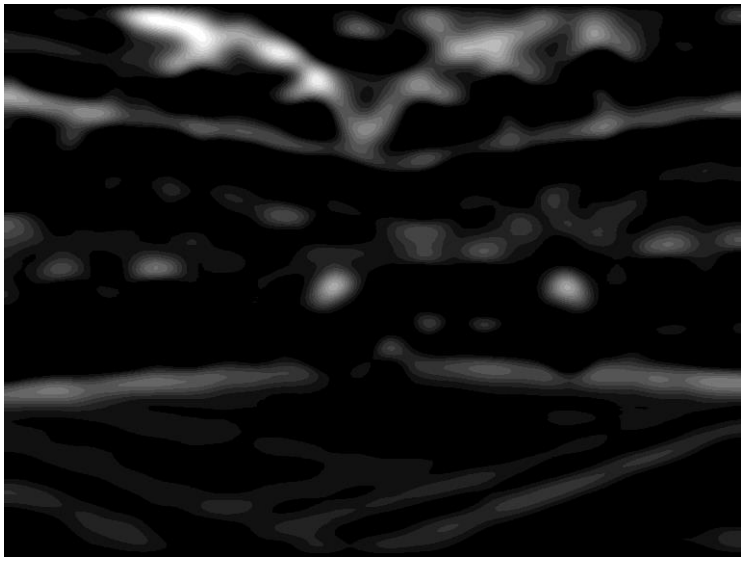
Derivative y image at k =8

Derivative x image at k =9

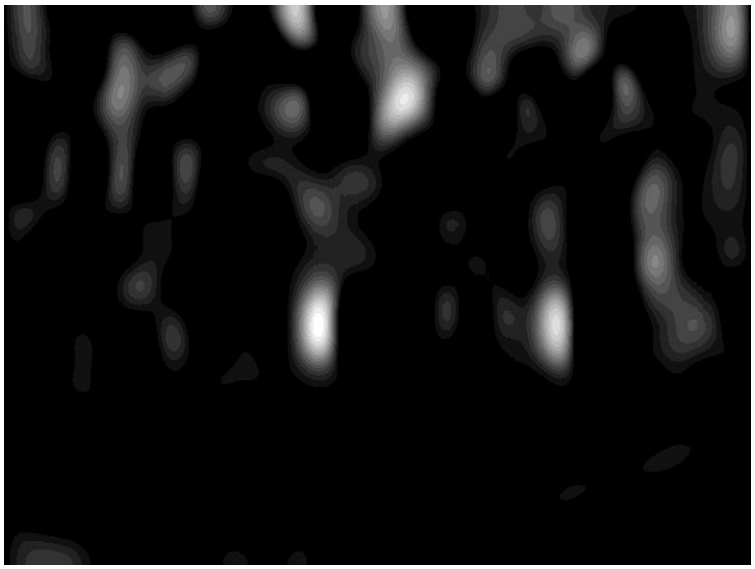


Derivative y image at k =9

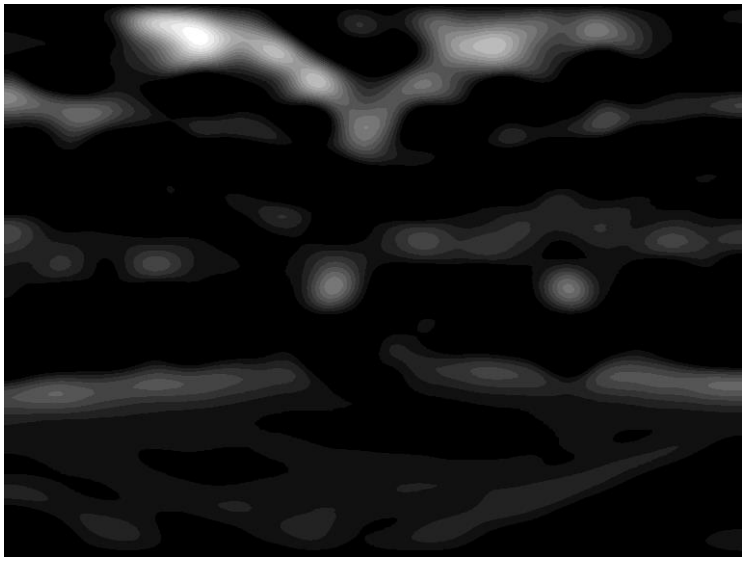
Derivative x image at k =10



Derivative y image at k =10



Derivative x image at k =11

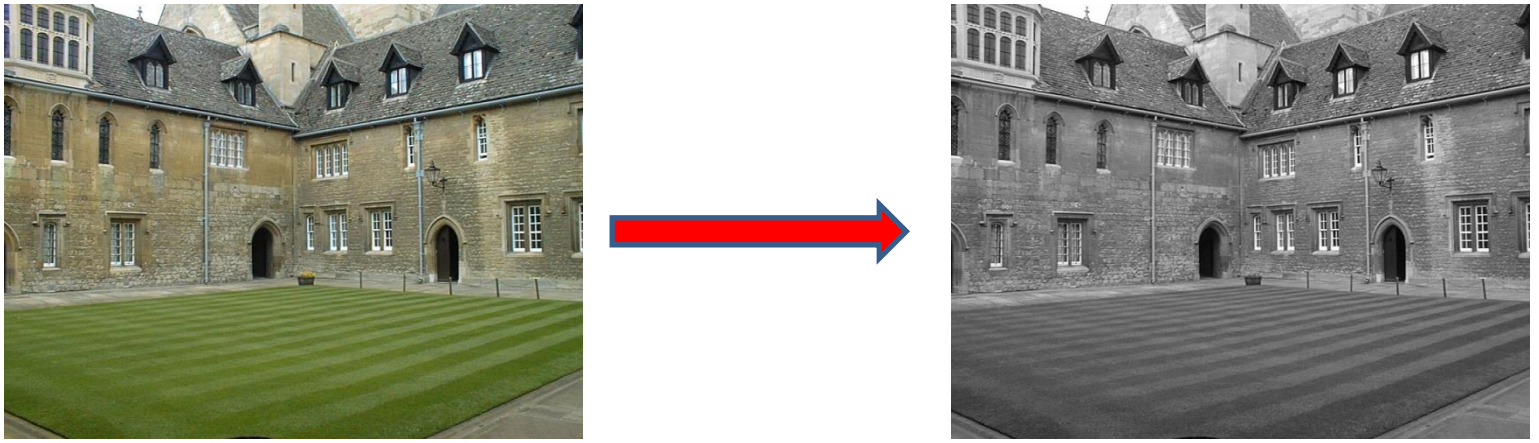


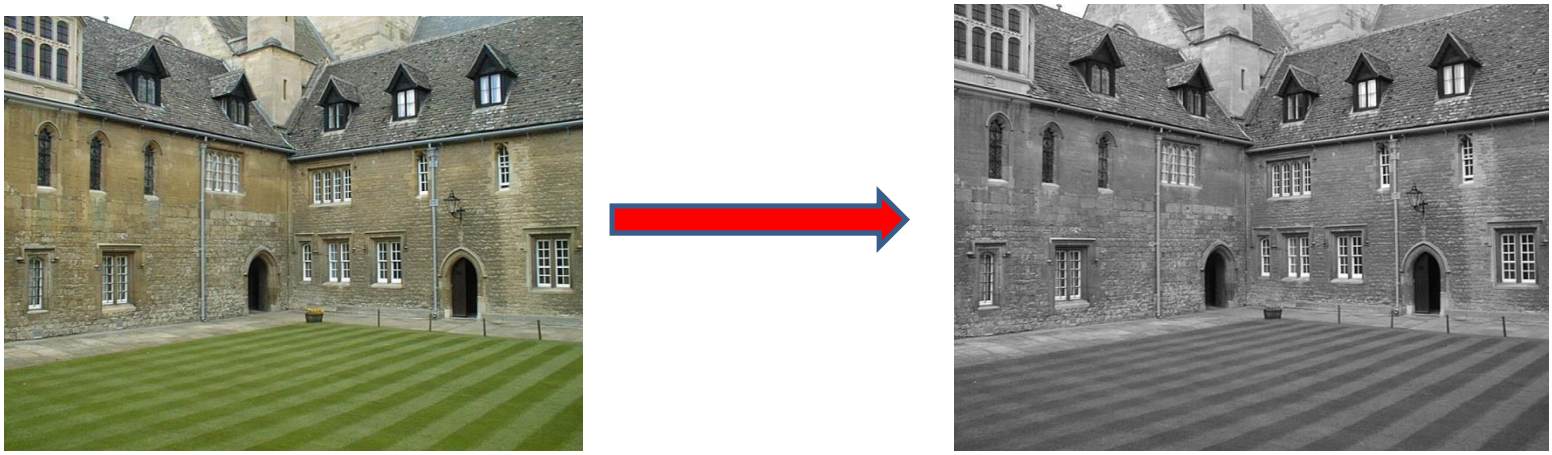
Derivative y image at k =11

Task 2 – Image Matching

Part (A): Download the input image files **Assignment_MV_01_image_1.jpg** (the same as in the previous task) and **Assignment_MV_01_image_2.jpg** from Canvas. Load both files and convert them into a single channel grey value image [*2 points*]. Make sure the data type is float32 to avoid any rounding errors [*1 point*].

Solution (A): We have converted both the image 1 and image 2 to the respective gray value image.



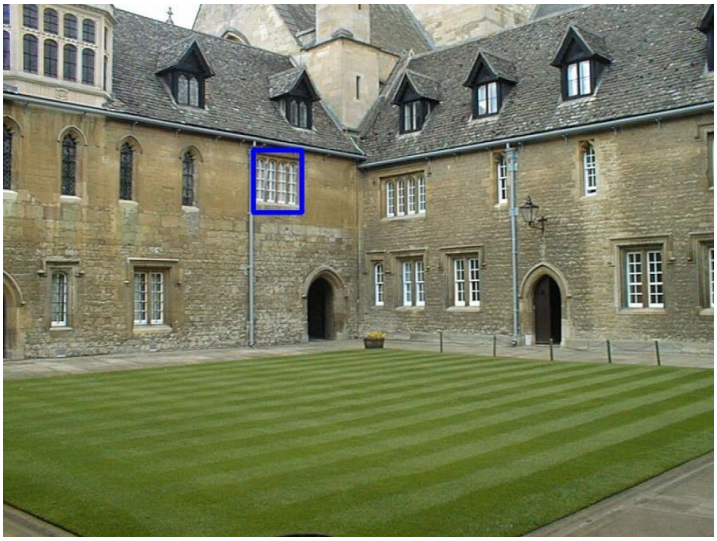


We have also made sure to convert the data type of both the images to float32.

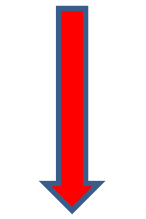
```
In [5]: runfile('C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/Assignment 1/
R00182615_MV_A01.py', wdir='C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/Assignment 1')
The data type of the first image before conversion is uint8
The data type of the second image before conversion is uint8

The data type of the first image after conversion is float32
The data type of the second image after conversion is float32
```

Part (B): The window on the 1st floor above the arch on the left wing is in a rectangle with the image coordinates ((360,210), (430,300)) in the first input image. Draw a rectangle around this window in the input image and display it [*1 point*]. Now cut out the image patch only containing the window and display it as image [*2 points*].

Solution (B): We have drawn a rectangle around the window with the image coordinates ((360,210), (430,300)) in the first input image and the resulting images are displayed as below. We have performed this task for the both the RGB image and the gray value image.



Now we have cut out the image patch from the rectangle image which only contains the window and is displayed as below.



Part (C): Calculate the mean and standard deviation of the cut-out patch from subtask B [*2 points*]. Go through all positions in the second input image and cut out a patch of equal size [*2 points*]. Also calculate mean and standard deviation and from this the cross-correlation between the two patches [*3 points*]. Create and display an image of all cross-correlations for all potential positions in the second image [*2 points*]. Find the position with maximum cross-correlation and draw a rectangle around this position in the second input image. Display the result [*2 points*].

Solution (C): The mean and standard deviation of the cut-out patch from the subtask B is

```
In [2]: runfile('C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/Assignment 1/
R00182615_MV_A01.py', wdir='C:/Users/sid/Desktop/CIT/Semester 2/Machine Vision/Assignment
1')
Mean of the cut-out patch is 161.46253967285156

Standard deviation of the cut-out patch is 60.22587203979492
```